IMPLEMENTASI DAN ANALISA PERFORMANSI PROTOKOL KEAMANAN TINYSEC PADA *WIRELESS SENSOR NETWORK* DENGAN MEDIA PENGIRIMAN DATA NRF24L01 MELALUI FREKUENSI RADIO

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Rafi Fajar Hidayat
NIM: 145150301111078

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI DAN ANALISA PERFORMANSI PROTOKOL KEAMANAN TINYSEC PADA *WIRELESS SENSOR NETWORK* DENGAN MEDIA PENGIRIMAN DATA NRF24L01 MELALUI FREKUENSI RADIO

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Rafi Fajar Hidayat
NIM: 145150301111078

Skripsi ini telah diuji dan dinyatakan lulus pada
19 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T., M.Eng.	Ari Kusyanti, S.T., M.Sc.
NIP: 19820809 201212 1 004	NIK: 201102 831228 2 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 19 Januari 2018

Rafi Fajar HIdayat

NIM: 145150301111078

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Implementasi dan Analisa Performansi Protokol Kemanan TinySec pada *Wireless sensor network* dengan Media Pengiriman Data NRF24L01 Melalui Frekuensi Radio" ini dapat terselesaikan. Laporan skripsi ini disusun dalam rangka memenuhi persyaratan untuk memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer

Penulis menyadari bahwa penyusunan laporan skripsi ini tidak lepas dari bantuan berbagai pihak baik secara langsung maupun tidak langsung. Dalam kesempatan ini penulis ingin menyampaikan ucapan terima kasih atas bantuannya kepada semua pihak, sehingga penulis dapat menyelesaikan laporan ini dengan baik. Ucapan terima kasih tersebut khususnya kepada :

1. Allah yang Maha Esa yang selalu memberikan petunjuk dan hikmah dalam penulisan ini.
2. Orang Tua dan Keluarga atas nasehat, kasih sayang serta dukungan materil dan moril.
3. Sabriansyah Rizqika Akbar, S.T., M.Eng. dan Ari Kusyanti, S.T, M.Sc. selaku dosen pembimbing yang telah memberikan banyak dukungan dalam proses penyusunan skripsi ini baik teknis maupun non teknis.
4. Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku ketua jurusan Teknik Informatika.
5. Sarah Anindya Shofi atas dukungan serta kesediaan untuk meluangkan waktu, motivasi dan juga doanya sampai pengerjaan skripsi selesai.
6. Teman-teman The-EX atas dukungan serta kesediaan untuk menjadi teman diskusi selama pengerjaan skripsi.
7. Dan orang-orang yang selalu mendukung serta mendoakan kelancaran proses skripsi ini yang tidak bisa disebutkan satu per satu atas semua doa dan dukungannya.

Dengan segala keterbatasan pengetahuan yang dimiliki, penulis sadar bahwa penulisan laporan skripsi ini masih jauh dari kesempurnaan. Sehingga penulis berharap agar laporan skripsi ini dapat bermanfaat dan merupakan salah satu informasi yang berguna bagi pembaca.

Malang, 19 Januari 2018

Penulis

Rafifajar22@gmail.com

ABSTRAK

Wireless Sensor Network (WSN) merupakan seperangkat jaringan yang terdiri dari beberapa *node* yang saling berkolaborasi untuk melakukan sensing, memproses, serta bertukar data dengan keterbatasan yang dimiliki dari tiap *node*. Isu terkait tantangan dan kendala yang ada pada WSN, salah satunya adalah security, isu security ada disebabkan oleh model komunikasi WSN yang dilakukan secara nirkabel. Pada penelitian kali ini merapkan protokol keamanan TinySec dengan mode autheticated *encryption* dan basis algoritma enkripsi RC5 dan CBC-mode. Perubahan tingkat performansi menjadi hal yang diperhatikan pada penelitian ini, serta keterbatasan yang ada pada media WSN yang digunakan, yaitu Mikrokontroler berbasis ATmega328p pada penelitian ini menggunakan arduino nano dan media komunikasi NRF24L01 melalui radio frekuensi 2.4GHz ISM. Penelitian ini terdiri dari 3 *node*, yaitu *node* pengirim, *gateway*, dan *node*. Dimana dalam penelitian ini dirancang dan diimplementasikan agar *node* pengirim dapat melakukan akuisisi data sensor menggunakan sensor udara dan data dapat di enkripsi pada *node* pengirim, selanjutnya dikirim kepada *node gateway* dan data dapat di dekripsi oleh *node gateway*. *Node attacker* bertugas untuk mengetahui tingkat keamanan proses pengiriman, sehingga melakukan *listening* data yang dikirimkan *node* pengirim secara *broadcast*. Dari hasil perancangan dan implementasi menghasilkan bahwa proses enkripsi, dekripsi, dan dokumentasi dapat dilakukan dengan baik, dan proses *listen node attacker* hanya mendapatkan data *ciphertext*.

Kata kunci: WSN, TinySec, Keamanan, RC5, CBC-mode, Arduino

ABSTRACT

Wireless Sensor Network (WSN) is a set of networks consisting of several collaborating nodes for sensing, processing, and exchanging data with the limitations of each node. Issues related to the challenges and obstacles that exist in the WSN, one of them is security, security issues exist caused by the communication model WSN is wireless. In the present study this involves the TinySec security protocol with authentication encryption mode and based of RC5 and CBC-mode encryption algorithms. The change of performance level has become a concern in this research, and the limitations of WSN media used, that is ATmega328p based microcontroller in this research using arduino nano and communication media NRF24L01 through ISM 2.4GHz frequency radio. This research consists of 3 nodes, namely sending node, gateway, and node. Where in this research is designed and implemented so that sending node can do the acquisition of sensor data using air sensor and data can be encrypted at sending node, then sent to gateway node and data can be decrypted by node gateway. Node attacker to know the security level of the sending process, so do listen to the data sent sending node attacker. From the design and implementation of encryption, decryption, and documentation process can be done well, and the process of node to node attacker only ciphertext data.

Keywords : WSN, TinySec, Security, RC5, CBC-mode, Arduino

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Data yang didapatkan melalui pengguna ataupun lingkungan akan dibentuk menjadi sebuah Informasi dengan sistem komputasi. Komputasi yang dimiliki oleh komputer saat ini memiliki berbagai fungsi, bermula dari melakukan komputasi data untuk mendapatkan informasi, hingga saat ini berbagi data dan atau informasi melalui konektivitas antar komputer dengan jaringan internet. Jaringan internet terbentuk melaui kabel dan tanpa kabel, jumlah perangkat yang terhubung dengan jaringan internet yang terus berkembang. Jumlah perangkat yang terhubung pada tahun 2010 melebihi jumlah populasi manusia di dunia yaitu dengan perbandingan 1,8 : 1, hingga tahun 2015 perbandingan jumlah perangkat yang terhubung dengan populasi manusia telah mencapai 3,5 : 1 (Evans, 2011). Pada tahun 2008 sebuah teknologi yang hingga saat ini juga terus berkembang, yaitu *Wireless Sensor Network* (WSN), WSN merupakan seperangkat jaringan yang terbentuk dari beberapa *node* yang saling berkolaborasi untuk melakukan *sensing*, memproses, serta bertukar data dengan keterbatasan yang dimiliki dari tiap *node*. WSN yang menggunakan media komunikasi nirkabel, dengan sistem *nodeing*, hal tersebut membuat WSN lebih rentan dibandingkan sistem yang menggunakan media komuniaksi kabel (Lighfoot, et al., 2007). WSN sendiri memiliki 4 *layer* L1-sensor, L2-*node*, L3-hub, dan L4-cloud. *Layer* 1 dan *Layer* 2 yang merupakan satu kesatuan berupa *Wireless sensor node*, yang terdiri dari beberapa *layer*, salah satunya data *link layer* (Laubhan, et al., 2016).

Dalam pengembangan WSN banyak isu terkait tantangan dan kendala yang dihadapi, seperti *energy*, *self-management*, *wireless networking*, *decentralized management*, *design constraints*, dan *security*. Kendala pengembangan WSN di bidang *security* yang disebabkan oleh WSN yang menerapkan sistem pengoperasian secara jarak jauh dan nirkabel. Kendala di bidang tersebut perlu diselesaikan karena WSN melakukan pemngambilan data yang memiliki tingkat sensitifitas tinggi (Dr Xuemin Shen, 2010). Berdasarkan hal di atas, maka diperlukan solusi sehingga WSN memiliki kapabiliitas yang sesuai dengan kebutuhan sistem. Mengimplementasikan protokol keamanan TinySec merupakan satu dari seluruh solusi yan ada dari permasalahan tesebut. TinySec merupakan protokol keamanan yang diimplementasikan pada *link-layer*, dengan jaringan nirkabel seperti 802.11b, GSM, dan radio. WSN yang memiliki keterbatasan sumber daya energi membutuhkan komponen pendukung yang sesuai dengan spesifikasi yang dimiliki, oleh karena itu penggunaan NRF24l01 sebagai media jaringan nirkabel dengan memanfaatkan frekuensi radio 2,4GHz dengan sifatnya yang *Ultra Low Power* (ULP) dan sesuai dengan arsitektur yang dimiliki protokol keamananan TinySec yang bersifat ringan dalam penerapannya, sehingga dapat menyelesaikan isu terkait tantangan dan kendala pengembangan WSN pada bidang *security*.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dijelaskan, rumusan masalah yang dapat di rumuskan pada penelitian ini antara lain:

1. Bagaimana menerapkan protokol keamanan TinySec pada *Wireless Sensor Network* dengan media pengiriman data NRF24l01 melalui frekuensi radio?
2. Bagaimana perbandingan tingkat perubahan performansi dari *Wireless Sensor Network* yang telah menerapkan protokol keamanan TinySec?

1.3 Tujuan

Adapun maksud dan tujuan dari penelitian pada Fakultas Ilmu Komputer Program Studi Teknik Komputer Universitas Brawijaya Malang ini adalah dapat menerapkan protokol keamanan TinySec pada *Wireless Sensor Network* dengan media pengiriman data NRF24l01 melalui frekuensi radio untuk menyelesaikan isu terkai tantangan dan kendala pengembangan WSN pada bidang *security*.

1.4 Manfaat

Pada jangka panjang, hasil penelitian ini dapat diimplementasikan untuk membuat sistem yang memberikan jaminan keamanan data pada setiap perangkat *Wireless Sensor Network* tanpa mengurangi tingkat efektifitas penggunaan *energy*. Meningkatkan performa teknologi *wireless sensor network* dan dapat diterapkan secara global sehingga bermanfaat dalam kehidupan sehari-hari. Sistem ini sebagai sarana peneliti dalam meningkatkan keilmuan di bidang Teknologi Informasi dan Ilmu Komputer. Serta diharapkan dapat menjadi bahan referensi untuk penelitian selanjutnya.

1.5 Batasan masalah

Batasan masalah ditentukan agar permasalahan yang dirumuskan dapat lebih terfokus dan tidak meluas. Pada penelitian ini, batasan masalah tersebut antara lain:

1. Sistem diterapkan pada mikrokontroler berbasis ATmega328p yaitu Arduino Nano.
2. NRF24l01 merupakan modul komunikasi yang diterapkan pada penelitian ini yaitu dengan mode komunikasi satu arah.
3. *Sample* performansi pada *Wireless Sensor Network* hanya difokuskan pada waktu delay yang dibutuhkan pada proses penggunaan protokol keamanan TinySec.
4. Protokol kemanan TinySec yang digunakan terbatas pada mode *Authenticated encryption* dan mengimplementasikan sistem *confidentiality* tanpa memperhatikan sistem *integrity* serta *authenticity* dengan menggunakan basis algoritma Enkripsi RC5 dengan mode operasi *Cipher Block Chaining* (CBC).
5. Penggunaan algoritma Enkripsi RC5 dengan total *word* 32 bit, total *key* 16 bit, dan total *round* 6.

1.6 Sistematika pembahasan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Bab ini menjelaskan dan memaparkan tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Bab ini membahas tinjauan pustaka dan dasar teori yang mendukung dalam pembuatan sistem pada penelitian ini. Pada bab ini juga terdapat penelitian-penelitian yang telah dilakukan sebelumnya sehingga dapat menjadi dasar pembuatan sistem ini.

BAB III Metodologi Penelitian

Bab ini membahas tentang langkah-langkah dalam melakukan penelitian ini, antara lain studi literatur, pengumpulan data, analisa kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian, dan evaluasi sistem.

BAB IV Rekayasa Kebutuhan

Bab ini menjelaskan secara keseluruhan kebetuhan sistem yang diperlukan meliputi kebutuhan perangkat keras, kebutuhan perangkat lunak, kebutuhan fungsional.

BAB V Perancangan dan Implementasi

Bab ini membahas tentang proses implementasi sistem dalam menerapkan protokol keamanan TinySec pada *Wireless Sensor Network* berbasis ATmega328p dengan media pengiriman data NRF24l01 melalui frekuensi radio. Proses dimulai dari perancangan dan implementasi perangkat keras serta implementasi perangkat lunak.

BAB VI Pengujian

Bab ini menjelaskan tata cara pengujian dan hasil analisis dari pengujian sistem.

BAB VII Penutup

Bab ini berisi tentang kesimpulan yang diperoleh dari perancangan, implementasi dan pengujian sistem pada penilitian ini, serta saran-saran untuk pengembangan penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi landasan kepustakaan yang berisi kajian pustaka dan dasar teori dalam menunjang proses penelitian. Kajian pustakan ini membahas terkait penelitian-penelitian yang telah dilakukan sebelumnya ada dan mendukung dalam proses penelitian yang diusulkan. Dasar teori ini berisi berbagai teori serta materi yang diperlukan dalam penelitian ini.

2.1 Kajian Pustaka

Kajian pustaka menjelaskan penelitian terkait dilakukan dengan menganalisis penelitian yang telah dilakukan sebelumnya serta memiliki kemiripan topik pembahasan dengan penelitian yang sedang dilakukan.

Tabel 2.1 Penelitian Terkait

No.	Penulis, Judul Penelitian	Metode	Hasil	Keterangan
1.	Ryandito Danuansa (2017) **Analisis Performansi dan Simulasi Protokol Keamanan Tinysec dan LLSP pada *Wireless* Sensoor Network**	Simulasi menggunakan aplikasi NS-3 serta *library* eksternal Crypto++ penerapan protokol kemanan TinySec dan LLSP	Kedua protokol mampu menyimpan *plaintext* dan dapat menjamin kerahasiaan data dalam jaringan, namun penerapan AES pada LLSP memiliki performa yang lebih baik dibandingkan dengan RC5 pada TinySec. Kedua protokol dapat memilah antara paket yang berasal dari *node* asli dengan *node* asing. Penggunaan protokol keamanan LLSP memiliki total penggunaan energi yang lebih rendah dibandingkan dengan TinySec	Kelebihan pada penelitian tersebut mampu membandingkan tingkat performasi 2 protokol TinySec dan LLSP Kekurangan pada penelitian tersebut, masih menggunakan simulasi

2.	Aditya Bhagus Aria Hutomo (2017) **Implementasi dan Analisa Perbandingan Performa *Wireless Sensor Network* Protokol Keamanan TinySec dan SPINS**	Simulasi penerapan protokol keamanan TinySec dan SPINS pada *Wireless Sensor Networ*	Protokol keamanan TinySec dan SPINS memiliki tingkat kesuksesan 100% dalam mengatasi serangan packet injection. Ukuran paket yang berbeda antara protokol kemananan Tinysec dengan SPINS memiliki perbedaan pada *key* dan digestnya. Dimana totan *key* TinyeSec memiliki total 10 byte dan SPINS 16 byte, serta digest pada TinySec 0 byte dan SPINS 16 byte. Untuk penggunan energi protokol lebih banyak menmbutuhkan energi dibandingkan TinySec.	Kekurangan pada penelitian tersebut,masih menggunakan simulasi

2.2 Dasar Teori

Dasar teori membahas berbagai teori dasar untuk menunjang proses penilitian. Pada subbab dasar teori akan dijelaskan referensi dan teori dasar pendukung mengenai gambaran umum tentang *Wireless Sensor Network*, serta kebutuhan *security system* pada teknologi *Wireless Sensor Network*, dan sistem pada protokol keamanan TinySec yang dapat diterapkan pada sistem *Wireless Sensor Network* dengan mikrokontroler berbasis ATmega 328p serta menerapkan media komunikasi frekuensi radio pada NRF24L01.

2.2.1 *Wireless Sensor Network*

Wireless Sensor Network (WSN) adalah sebuah jaringan yang terdiri dari sekumpulan *node-node* sensor yang memiliki kemampuan untuk mendeteksi suatu keadaan di sekitarnya, memproses data yang diperoleh, dan saling berkomunikasi satu sama lain untuk mengirimkan data ke *node* pusat (Sen, 2009). Pada umumnya WSN digunakan untuk memantau keadaan suatu lingkungan, seperti suhu, suara, tekanan, dll. WSN mulai banyak diterapkan karena beberapa keunggulannya, antara lain yaitu rendah daya atau energi, skalabilitas yang baik, mudah diimplementasikan, ukuran yang kecil, dan sebagainya. Tetapi WSN juga

memiliki masalah pada keamanan dan rentan terhadap serangan yang dilakukan pada jaringan, seperti *sniffing attack* atau *man-in-the-middle*. Untuk menangani masalah tersebut dapat dilakukan dengan mengimplementasikan protokol keamanan. Namun karena ukurannya yg kecil WSN memiliki keterbatasan *resource* atau sumber daya (seperti memori, energi, *processor*), sehingga untuk melakukan proses komputasi yang kompleks akan membutuhkan energi yang besar pula. Hal ini lah yang menjadi permasalah utama dalam WSN. Untuk itu protokol keamanan dituntut untuk dapat memberikan kemanan yang baik namun dengan konsumsi energi yang rendah.

2.2.2 Security Requirements

Tantangan utama dalam penerapan keamanan *Wireless Sensor Network* disimpulkan ke dalam beberapa *point-point* berikut :

1. Keterbatasan sumber daya memori

   Jumlah penyimpanan pada memori utama sangat terbatas pada setiap *node*, sehingga menyebabkan kurangnya sumber daya untuk membuat serta menyimpan *key* unik dari tiap *node* pada satu jaringan.

2. Keterbatasan *bandwith* dan *transmission power*

   Sensor network memiliki salah satu ciri khas yaitu memiliki bandwith yang sangat lambat. Salah satu contohnya, UC Berkeley Mica yang hanya memiliki kecepatan bandwith sebesar 10 Kbps, dengan ukuran data hingga 30 Bytes. Tingkat kehandalan yang rendah, membuat proses komunikasi dengan ukuran data yang besar menjadi tidak layak.

2.2.3 Security Goals

Wireless Sensor Network memiliki 3 *security goals* yang harus dipenuhi dalam penerapa protokol keamanan, 3 *security goals* yang harus dipenuhi data *authentication*, data *integrity*, dan data *confidentiality*

1. Data *Authetication*

Komunikasi di antara *Wireless Sensor Network*, menerapkan 2 sistem *authentication, node authetication* dan data *authentication*. Sistem *autentication* yang diterapkan bertugas untuk mengatur akses dan mencegah *node* tidak dikenal pada jarigan yang sama. *Node* yang mengatur pemberian akses harus memiliki kapabilitas untuk mendeteksi peran dari *node* tidak dikenal dan menolaknya. Data *authentication* berfungsi untuk memastikan tingkat integritas dari data pada pesan untuk diverifikasi (Ren, et al., 2005).

2. Data *Confidentiality*

Data *confidentiality* memiliki tujuan untuk menjaga kerahasiaan informasi dari *node* yang tidak dikenal. Data *confidentiality* dihasilkan dari proses enkripsi data, yang menghasilkan *key* rahasia bersama yang hanya akan

dimiliki oleh penerima data. Hasil data enkripsi berupa *chiper* yang memiliki tingkat kekuatan tinggi, sulit untuk dipecahkan oleh *node* yang tidak dikenal pada satu jaringan (Ren, et al., 2005).

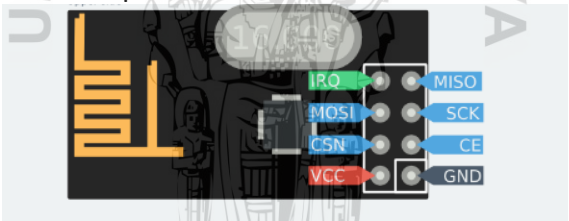
3. Data *Integrity*

Data *integrity* memiliki tujuan untuk membedakan antara data yang berasal dari *node* yang tidak dikenal dengan *node* yang bertugas mengirimkan data. Data *integrity* akan menambahkkan *message authentication code* pada paket data, sehingga paket data yang dikirimkan *node* asli akan berbeda dengan data yang dikirimkan *node* yang tidak dikenal.

2.2.4 NRF24L01

Wireless Module NRF24L01 merupakan modul komunikasi nirkabel yang memanfaatkan pita gelombang RF 2.4GHz ISM (*Industrial, Scientific and Medical*). Modul komunikasi ini menggunakan *serial peripheral interface* (SPI) untuk berkomunikasi. Tegangan kerja dari modul ini adalah 3.3V DC. NRF24L01 memiliki *baseband logic Enhfanced ShockBurst™ hardware protocol accelerator* yang support "*high-speed* SPI *interface for the application controller*". NRF24L01 memiliki *true ULP solution*, yang mendukung ketahanan penggunaan daya baterai. Modul ini terdri dari 8 buah pin seperti pada gambar 2.1 dibawah ini :

Berikut adalah gambaran dan bagian pin-pin dari modul komunikasi NRF24l01 yang tediri dari 8 pin antara lain :



Gambar 2.1 Konfigurasi Pin NRF24L01
Sumber: (www.sunrom.com, 2017)

Spesifikasi secara detail yang dimiliki oleh modul NRF24L01 dijelaskan pada tabel berikut :

Tabel 2.2 Spesfikasi *Module* NRF24L01

Power supply	1.9V~3.6V
I/O port working voltage	0~3.3, 5 V
I/O port working current	13.5mA at 2Mbps
Data rate	256kbps / 1Mbps / 2Mbps
Receiving sensitivity	-85dBm at 1Mbps
Transmission range	70~100 meter at 256kbps
Temperatures	Operating:-40°C ~ 85°C / Storage:-40°C ~ 125°C

Sumber: (www.dx.com, 2016)

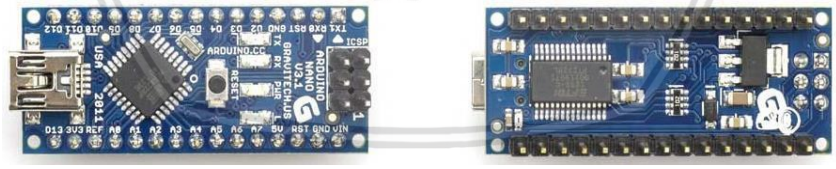
2.2.5 Mikrokontroler

Mikrokontroler adalah sebuah sistem mikroprosesor dimana didalamnya telah dilengkapi oleh *Read Only Memory (ROM), Random Accsess Memory (RAM),CPU, Input-Output, timer, interrupt, Clock* dan komponen internal lainnya yang telah terhubung dan terorganisasi dengan baik pada satu chip yang siap diimplementasikan.

Gambar 2.2 Atmega328P
Sumber : (Microchip, 2017)

ATmega328P memiliki spesifikasi detail, yaitu memiliki 8 Kb system programmable flash dengan kemampuan *read while write*, 1 KB EEPROM, 2 KB SRAM, 23 *general purpose* I/O, 32 register berbagai fungsi, 3 *timer/counter*, Interrupt internal maupun eksternal, jalur serial untuk pemrograman dengan menggunakan USART, *serial peripheral interface* (SPI), *two wire interface* (I2C), 6 port *Pulse Width Modulation* (PWM), 6 port 10 bit ADC dan Watchdog Timer dengan osilator internal.

Mikrontroler berbasis ATmega328p yang digunakan pada penelitian ini adalah Arduino Nano v3.0. Arduino merupakan *developer board* yang dijual secara komersil sehingga mudah untuk didapatkan. Arduino Nano versi 3.0 merupakan bentuk pengembangan arduino nano. Arduino nano memiliki *body* yang relatif kecil jenis *breadboard-friendly board* dengan mikroprosesor ATmega328 pada arduino nano versi 3. Pada gambar 2.3 memperlihatkan gambaran bentuk yang dimiliki arduino nano versi 3.0 dan pada Tabel 2.3 menjelaskan spesifikasi yang dimiliki arduino nano versi 3.0.



Gambar 2.3 Arduino Nano V3.0

Sumber: (https://www.arduino.cc/en/Main/Arduino*Board*Nano)

Tabel 2.3 Arduino Nano V3.0

Mikrokontroler	Atmel ATmega328P
Operating Voltage	5 V
Input Voltage	7-12 V

Digital I/O Pins	14 (of which 6 provide PWM *output*)
Analog *Input* Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz

Sumber : (https://www.arduino.cc/en/Main/Arduino*Board*Nano)

2.2.6 Protokol Keamanan TinySec

TinySec merupakan protokol keamanan yang diterapkan pada *link layer* dengan menyesuaikan arsitektur *Wireless Sensor Network*. TinySec didesain untuk dapat diterapkan pada sistem *wireless* seperti 802.11b, GSM, dan radio. Dibandingkan dengan protokol keamanan seperti IPSec, SSL/TLS, dan SSH yang memberikan jaminan atas keamanan komunikasi melalui jaringan internet, namun membutuhkan sumber daya yang besar jika diterapkan pada *sensor network*. TinySec menggunakan 2 tipe keamanan, TinySec-AE untuk melakukan *Authenticated encryption* dan TinySec-Auth yang hanya melakukan *Authenticated*. Dalam memenuhi kebutuhan *confidentiality* dalam protokol keamanan TinySec menggunakan enkripsi dengan algoritma enkripsi berbasis *Block cipher*. *Block cipher* yang telah diterapkan sebelumnya yaitu menggunakan algoritma RC5 serta Skipjack sebagai algoritma yang sesuai dengan kebutuhan dalam implementasi pada mikrokontroler, serta menambahkan mode operasi Ciper *Block* Chaining sebagai penambahan parameter untuk meningkatkan daya tahan terjadinya kebocoran informasi.

2.2.7 Algoritma RC5

RC5 merupakan algoritma enkripsi berbasis symmetric *Block cipher* yang cocok untuk diterapkan pada *hardware* dan software, dengan sistem operasi komputasi primitif yang digunakan pada beberapa mikroprosessor. RC5 memiliki sifat *word* oriented, semua operasi komptasi dasar memiliki *u*-bit sebagai *input* dan *output*. RC5 merupakan *Block cipher* dengan dua *word-input* sebagai *plaintext block size* dan dua *word-output* sebagai *ciphertext block size*. RC5 memiliki 3 jenis total bits *word* yaitu 16, 32, dan 64. Selain menetukan total *word* (*w*) yang digunakan, ada beberapa parameter lain yang juga dibutuhkan pada algoritma RC5, beberapa parameter tersebut akan dijelaskan pada tabel 2.6 berikut ini.

Tabel 2.4 Parameter pada algoritma RC5

Parameter	Keterangan
w	Panjang *word* dalam bits, dengan bebeberapa tipe 16, 32, atau 64 yang akan terdiri dari 2 wird *block*
u = w/8	Panjang *word* dalam bytes
b	Panjang *key* dalam bytes
K[]	*Key*, yang dirubah menjadi array bytes (menggunakan 0-based indexing)
c	Panjang *key* dalam *word*s (or1, if b = 0)
L[]	Temporary array yang digunakan dalam proses *key* sceduling
r	Total *round* dalam proses enkripsi dan dekripsi data
t	Total *round* sub*key* yang dibutuhkan
S[]	Sub*key word*s
Pw	Nilai *magic* konstan pertama yang telah ditentukan apa bila w = 16 pw = 0xB7E1, w = 32 pw = 0xB7E15163, w = 64 pw = 0xB7E151628AED2A6B
Qw	Nilai *magic* konstan kedua yang telah ditentukan apa bila w = 16 pw = 0x9E37, w = 32 pw = 0x9E3779B9, w = 64 pw = 0x 0x9E3779B97F4A7C15

Parameter yang ada pada tabel 2.5 dibutuhkan dalam proses yang dilakukan algoritma RC5, dalam prosesnya RC5 memiliki 3 bagian proses, yaitu *key expansion*, *encryption*, dan *decryption*. *Key expansion* merupakan proses melakukan ekspansi *key* rahasia K yang akan disipan pada array S, dengan total array sejumlah t dengan menggunakan kedua nilai *magic* konstan, proses pertama merubah *key* rahasia dari *word*ss menjadi bytes dengan algoritma pada gambar 2.7

$$c = \lceil \max(b, 1)/u \rceil$$
$$\text{for } i = b - 1 \text{ downto } 0 \text{ do}$$
$$L[i/u] = (L[i/u] \lll 8) + K[i];$$

Gambar 2.4 Proses merubah *key* rahasia dari *word*s menjadi bytes

Setelah merubah *key* rahasia dari *word*s menjadi bytes selanjutnya adalah proses menginisialisasi nilai dari array S, pada gambar 2.8 akan menampilkan algoritma proses menginisialisasi nilai dari array S.

$S[0] = P_w;$
for $i = 1$ to $t - 1$ do
$S[i] = S[i - 1] + Q_w;$

Gambar 2.5 Proses menginisialisasi nilai dari array S

Selanjutnya proses yang dilakukan adala melakukan mixing secret *key*, hal ini merupakan step terakir proses *key expansion*, yang akan melibatkan array S dan array L. Pada gambar 2.9 akan menampilkan algoritma proses mixing secret *key*.

$i = j = 0;$
$A = B = 0;$
do $3 * \max(t, c)$ times:
$A = S[i] = (S[i] + A + B) \lll 3;$
$B = L[j] = (L[j] + A + B) \lll (A + B);$
$i = (i + 1) \bmod(t);$
$j = (j + 1) \bmod(c);$

Gambar 2.6 Proses *mixing secret key*

Setelah proses *key expansion* dan menghasilkan total *key* sejumlah t proses yang dilakukan selanjutnya adalah proses enkripsi, yang akan melibatkan 2 w-bit register yang akan disimpan pada variabel A dan B, pada gambar 2.10 merupakan algoritma proses enkripsi.

$A = A + S[0];$
$B = B + S[1];$
for $i = 1$ to r do
$A = ((A \oplus B) \lll B) + S[2 * i];$
$B = ((B \oplus A) \lll A) + S[2 * i + 1];$

Gambar 2.7 Algoritma proses enkripsi

Proses enkripsi akan menghasilkan *ciphertext*, untuk dapat mengembalikan kebentuk *plaintext* dpat melakukan proses dekripsi, pada proses dekripsi juga di perlukan 2 w-bit, yang berasal dari *ciphertext* utama yang akan di bagi 2 dan disimpan pada variable A dan B, Pada gambar 2.11 merupakan algoritma dekripsi.

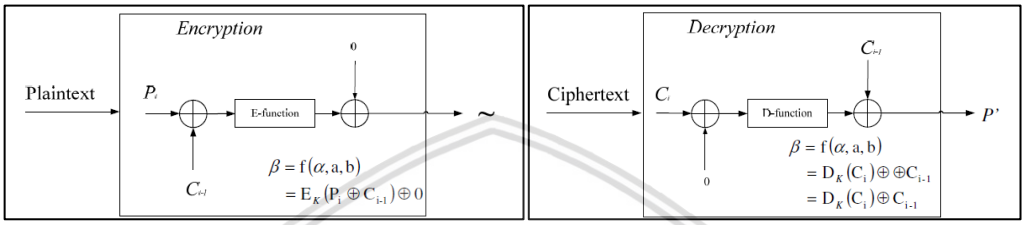
for $i = r$ downto 1 do
$B = ((B - S[2 * i + 1]) \ggg A) \oplus A;$
$A = ((A - S[2 * i]) \ggg B) \oplus B;$
$B = B - S[1];$
$A = A - S[0];$

Gambar 2.8 Proses algoritma dekripsi

Ketiga proses yang ada pada algoritma RC5, yaitu proses *key expansion*, enkripsi, dan dekripsi merupakan satu kesatuan prose yangterjadi dalam algoritm RC5.

2.2.8 Algoritma Chiper *Block* Chaining (CBC)

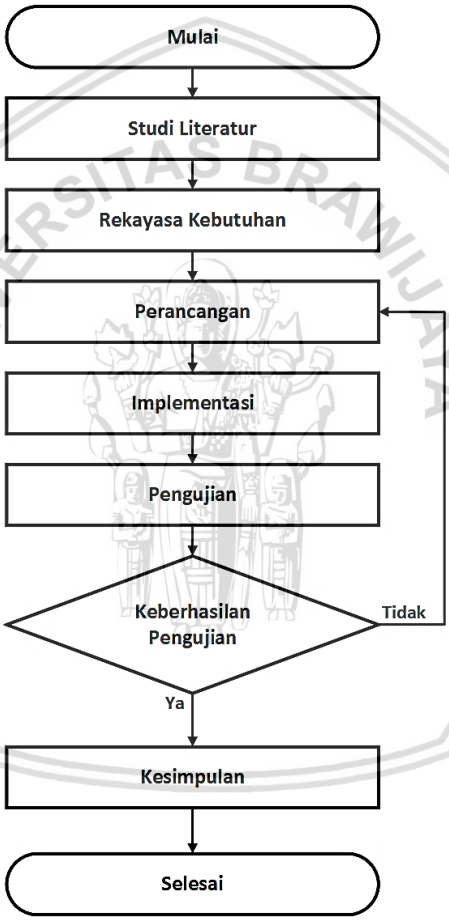
Cipher Block Chaining (CBC) merupakan *Block cipher* mode yang menyediakan proses *confidentiality* tetapi tidak untuk proses message *integrity* pada kriptografi. CBC mode, malukan XOR *block plaintext* dengan *ciphertext block* sebelumnya sbeellum melakukan enkripsi. Fungsi yang dimiliki CBC mode yaitu membuat pesan unik, hal ini di sebabkan dengan adanya *Initialization Vector* yang digunakan pada *block* pertama. Penggunaan CBC mode diterapkan pada dua proses enkripsi dan dekripsi, pada gambar 2.12 menampilkan proses enkripsi dan dekripsi pada CBC mode.



Gambar 2.9 Proses enkripsi dan dekripsi pada CBC mode

BAB 3 METODOLOGI PENELITIAN

Bab metodologi penelitian menjelaskan mengenai metodologi yang diterapkan pada penelitian. gambar 3.1 merupakan diagram alir yang berisi proses serta tahapan yang dilakukan pada metodologi penelitian Implementasi dan Analisa Performansi Protokol Keamanan TinySec pada *Wireless Sensor Network*. Tahapan tersebut dimulai dari studi literatur, rekayasa kebutuhan, perancangan, implementasi, pengujian dan analisis serta pengambilan kesimpulan dan saran. Langkah-langkah yang dilakukan dalam penelitian ini ditunjukkan pada gambar 3.1.



Gambar 3.1 Diagram Alir Metode Penelitian

3.1 Studi Literatur

Studi literatur merupakan tahapan pencarian literatur yang akan menunjang penyusunan dasar teori dalam sistem. Literatur tersebut diperoleh dari buku, jurnal, dan *website* terkait. Teori tersebut meliputi

1. *Wireless Sensor Network*

   Wireless Sensor Network merupakan sebuah jaringan yang terdiri dari sekumpulan *node-node* sensor yang memiliki kemampuan untuk mendeteksi suatu keadaan di sekitarnya, memproses data yang dieroleh, dan saling berkomunikasi satu sama lain untuk mengirimkan data ke *node* pusat.

2. *Security Requirements*

   Tantangan utama dalam penerapan keamanan *Wireless Sensor Network* memiliki 2 *point* utama, yaitu keterbatasan sumber daya memori dan keterbatasan bandwith dan transmission power.

3. *Security Goals*

   Wireless Sensor Network memiliki 3 security goals yang harus dipenuhi dalam penerapapn protokol keamanan, yaitu data *authentication*, data *integrity*, dan data *confidentiality*.

4. Mikrokontroler

   Mikrokontroler adalah sebuah sistem mikroprosesor dimana didalamnya telah dilengkapi oleh *Read Only Memory (ROM), Random Accsess Memory (RAM),CPU, Input-Output, timer, interrupt, Clock* dan komponen internal lainnya yang telah terhubung dan terorganisasi dengan baik pada satu chip yang siap diimplementasikan. Salah satunya adalah ATmega328P yang di aplikasikan ke beberapa *dev-board* seperti Arduino Nano.

5. Modul *Wireless* Nrf24l01

   Wireless Module NRF24L01 merupakan modul komunikasi nirkabel yang memanfaatkan pita gelombang RF 2.4GHz ISM (*Industrial, Scientific and Medical*). Modul komunikasi ini menggunakan *serial peripheral interface* (SPI) untuk berkomunikasi.

6. Protokol Keamanan TinySec

   TinySec merupakan protokol keamanan yang diterapkan pada *link layer* dengan menyesuaikan arsitektur *Wireless Sensor Network.* TinySec didesain untuk dapat diterapkan pada sistem *wireless* seperti 802.11b, GSM, dan radio.

3.2 . Analisis Kebutuhan

Analisis kebutuhan ditujukan untuk menganalisa kebutuhan yang akan mendukung dalam proses penelitian ini terutama dalam proses perancangan, impelementasi dan pengujian. Analisa kebutuhan pada penelitian ini antara lain :

3.2.1 Kebutuhan Perangkat Keras

1. Mikrokontroler berbasis ATmega328p (Arduino-UNO atau Arduino Nano)
2. Sensor berbasis analog (sensor udara MQ-2)
3. NRF24L01

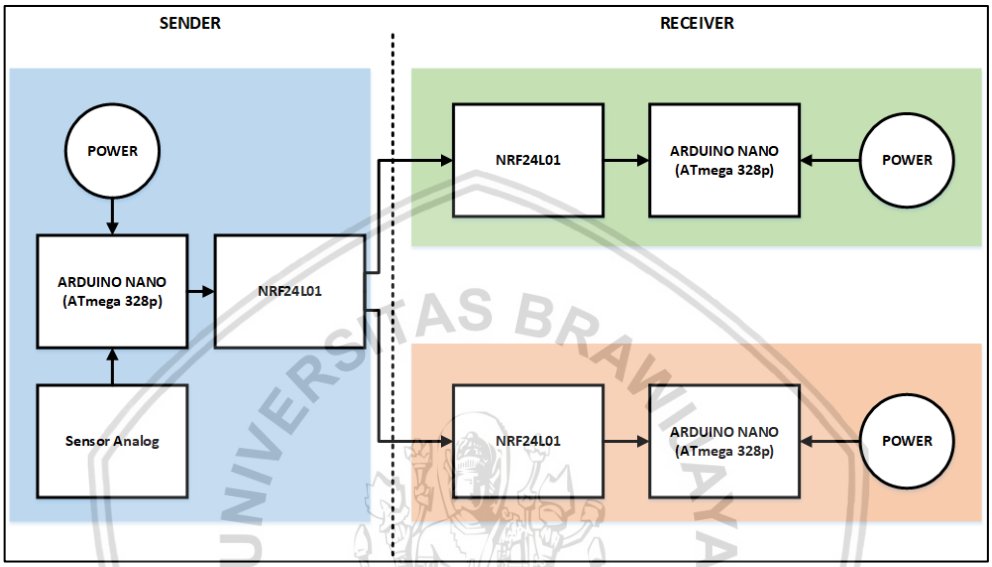
3.2.2 Kebutuhan Perangkat Lunak

1. Arduino IDE
2. RF24 *Library*

3. Protokol keamanan TinySec dengan basis algoritma RC5

3.3 . Perancangan Sistem

Perancangan merupakan tahapan yang dilakukan setelah seluruh kebutuhan dari sistem diperoleh dalam proses analisis kebutuhan. Perancangan sistem dimulai dengan membuat diagram blok. Diagram blok ini berupa perancangan yang dilakukan dalam *node-node* serta bagaimana *node* tersebut nantinya akan melakukan komunikasi dan melakukan pengiriman data.



Gambar 3.2 Diagram blok sistem

Keterangan :

: Diagram blok *node* pengirm

: Diagram blok *node gateway*

: Digaram blok *node attacker*

Berdasarkan Gambar 3.2 yang merupakan diagram blok keseluruhan sistem dapat dijelaskan sebagai berikut :

1. Komponen yang ada pada diagram blok sistem salah satunya adalah diagram blok *node* pengirim, *node* pengirim yang memiliki fungsi untuk melakukan sensing data, dilakukan dengan menggunakan sensor analog yang terhubung langsung dengan mikrokotroler arduino nano. Mikrokotroler arduino nano memiliki fungsi untuk memproses data melakukan enkripsi dan mengirim kan data menggunakan NRF24L01.

2. Diagram blok *node gateway*, merupakan perancangan dari *node gateway* yang memiliki fungsi untuk menerma data menggunakan NRF24L01 dan selanjutnya data di proses menggunakan mikrokontroler arduino nano untuk memlakukan proses dekripsi data.

3. Diagram blok *node attacker*, merupakan perancangan *node attacker* yang memiliki fungsi untuk menerima data menggunakan NRF24L01 namun data yang diterima melalui NRF24L01 dan di proses mikrokontroler arduino nano namun tidak dapat melakukan proses dekripsi.

4. Komunikasi yang dilakukan *node* pengirim, *node gateway* dan *node attacker* dilakukan dengan menggunakan media pengiriman data NRF24L01 dan dilakukan satu arah, *node* pengirim sebagai *sender* dan *node gateway* serta *node attacker* sebagai *receiver*.

3.4 . Implementasi

Implementasi berdasarkan pada hasil perancangan yang sebelumnya dilakukan. Tahapan implementasi yang dilakukan terdiri dari dua tahap yaitu, implementasi perangkat keras dan perangkat lunak.

3.4.1 Implementasi Perangkat Keras

Implementasi perangkat keras yang dilakukan meliputi implementasi perangkat keras pada *node* pengirim, *node gateway*, dan *node attacker*. Implementasi dimulai dengan spesifikasi sistem. Setelah itu sistem dirangkai agar dapat terhubung satu sama lain.

1. Implementasi node pengirim

Implementasi node pengirim dimulai dengan memenuhi spesifikasi sistem. Perangkat yang digunakan pada *gateway* adalah modul komunikasi nRF24L01, sensor udara MQ-2, dan mikrokotroler arduino nano. Modul nRF24L01, sensor udara mQ-2, dan mikrokontroler arduino nano dihubungkan untuk mengirimkan data.

2. Implementasi node *gateway*

Implementasi node *gateway* dimulai dengan memenuhi spesifikasi sistem. Perangkat yang digunakan pada *gateway* adalah modul komunikasi nRF24L01, dan mikrokotroler arduino nano. Modul nRF24L01, dan mikrokontroler arduino nano dihubungkan untuk menerima data.

3. Implementasi node *Attacker*

Implementasi node *Attacker* dimulai dengan memenuhi spesifikasi sistem. Perangkat yang digunakan pada *gateway* adalah modul komunikasi nRF24L01, dan mikrokotroler arduino nano. Modul nRF24L01, dan mikrokontroler arduino nano dihubungkan untuk menerima data.

3.4.2 Implementasi Perangkat Lunak

Implementasi perangkat lunak pada *node* pengirim, *node gateway*, dan *node attacker*.

1. Implementasi *node* pengirim

Implementasi *node* pengirim dimulai dengan mengimplementasikan kode program untuk melakukan proses enkripsi data menggunakan protokol keamanan TinySec dengan basis algoritma enkripsi RC5 serta penggunaan RF24 *library* untuk menerapkan komunikasi melalui NRF24L01 dengan *node gateway*.

2. Implementasi *node gateway*

Implementasi *node gateway* dimulai dengan mengimplementasikan kode program untuk melakukan proses dekripsi data menggunakan protokol keamanan TinySec dengan basis algoritma enkripsi RC5 serta penggunaan RF24 *library* untuk menerapkan komunikasi melalui NRF24L01 dengan *node* pengirim dan *node*.

3. Implementasi *node attacker*

Implementasi *node* pengiirim dimulai dengan mengimplementasikan RF24 *library* untuk menerapkan komunikasi melalui NRF24L01 untuk menerima data yang dikirimkan oleh *node* pengirim.

3.5 . Pengujian dan Analisis

Pengujian dan analisis ini dilakukan pada sistem agar dapat diketahui apakah sistem telah berjalan sesuai dengan spesifikasi dari rancangan yang telah dibuat. Pengujian yang dilakukan adalah sebagai berikut :

1. Pengujian dengan melakukan enkripsi dan pengiriman data *node* pengirim kepada *node gateway*.
2. Pengujian dengan melakukan penerimaan data dan dekripsi data pada *node gateway*.
3. Pengujian dengan melakukan penerimaan data hasil enkripsi *node* pengirim pada *node attacker*.

Dari pengujian yang telah dilakukan, selanjutnya akan dilakukan tahap analisis untuk mengetahui tingkat performansi penerapan perancangan dan spesifikasi yang telah dibuat.

3.6 . Kesimpulan

Kesimpulan didapatkan setelah melakukan tahap perancangan, implementasi, pengujian dan analisis terhadap sistem yang dibuat. Kesimpulan disusun berdasarkan hasil dari tahap pengujian dan analisis yang dilakukan pada sistem yang dibuat. Isi pada kesimpulan diharapkan dapat menjadi acuan dasar pada penelitian selanjutnya untuk mengembangkan penerapan protokol keamanan pada *Wireless Sensor Network*. Di akhir penulisan terdapat saran yang bertujuan untuk memberikan kemudahan penelitian selanjutnya, apabila akan meneruskan dan mengembangkan penelitian ini.

BAB 4 REKAYASA KEBUTUHAN

Dpada bab ini memberikan penjelasan mengenai persyaratan-persyaratan yang harus dipenuhi serta persyaratan lainnya yang bersifat opsional sehingga setiap fungsi dari sistem akan berjalan dengan baik serta sesuai dengan tujuan penelitian ini.

4.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan yang wajib dipenuhi agar sistem utama dapat berjalan sesuai dengan fungsinya. Kebutuhan fungsonal meliputi 2 bagian, yakni kebutuhan dari sisi pernagkat keras maupun dari sisi perangkat lunak. Bagian-bagian ini merupkan komponen penting dalam fungsionalitas sistem ini, sehingga 2 bagian ini wajib dipenuhi.

4.1.1 Kebutuhan Perangkat Keras

Pada bagian ini menjelaskan terkait dengan kebutuhan perangkat keras pada sistem ini. Perangkat keras ini meliputi dari sekumpulan komponen elektronika yang akan membentuk suatu sistem perangkat keras baru yang memiliki fungsi tertentu. Berikut adalah kebutuhan perangkat keras yang digunakan pada proses implementasi sistem, yakni sebagai berikut :

1. Mikrokontroler berbasis ATmega328p (Arduino-UNO atau Arduino Nano)
   Mikrokontroler Arduino Uno atau arduino nano memiliki chip berbasis Atmega328 p sehingga digunakan untuk menjalankan kode program dan terhubung dengan media komunikasi NRF24L01 menggunakan SPI serta pembacaan data sensor.
2. Sensor berbasis analog (sensor udara MQ-2)
   Sensor berbasis analog, berfungsi sebagai media akuisisi data pada *Wireless Sensor Network*, pada proses penelitian ini menggunakan sensor MQ-2 yang peka terhadap kadar asap pada udara, dengan hasil akuisisi data yang relatif konstan.
3. Modul NRF24l01
   Modul NRF24l01 merupakan modul komunikasi yang berfungsi untuk mengirimkan data dari mikrokontroler secara *wireless*. Menggunakan sinyal radio dengan frekuensi 2.4GHz. Jadi untuk penggunaan modul ini tidak memerlukan jaringan internet seperti wifi. Namun untuk melakukan kegiatan komunikasi diperlukan 2 buah modul NRF24l01 untuk bertindak sebagai *Transmitter* dan *Receiver*, yang berfungsi untuk mengirim dan menerima data pada tiap *node*.

4.1.2 Kebutuhan Perangkat Lunak

Pada bagian ini menjelaskan terkait dengan kebutuhan perangkat lunak pada sistem ini. Perangkat lunak ini meliputi kesuluran sub-sistem yang mewakili fungsionalitas sistem yang akan berjalan sesuai dengan perancangan sistem. Pemberian nomor kode dokumen kebutuhan perangkat lunak adalah 1000 sampai dengan 4003.

4.1.2.1 Mekanisme Enkripsi Data Protokol Keamanan TinySec *Node* Pengirim

REQ-ProTiny-Encrypt-1000 – Akuisisi data sensor					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `analogRead(Pin)` berfungsi untuk mendapatkan data dari sensor analog sehingga data yang telah di akusisi dapat dienkripsi.					
Keterangan:					

REQ-ProTiny-Encrypt-1001 – Insiasi *Initialization Vector*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini berfungsi untuk mendapatkan *Initialization Vector* (IV) yang digunakan pada algoritma CBC-mode					
Keterangan:					

REQ-ProTiny-Encrypt-1002 – Inisialisasi *key* dan ekspansi *key*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini berfungsi untuk mendapatkan total 14 *key* dengan panjang 16 bit tiap *key* sesuai dengan algoritma RC5					
Keterangan:					

REQ-ProTiny-Encrypt-1003 – Enkripsi data					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `Enkripsi()` berfungsi untuk melakukan enkripsi data yang telah diakuisisi, menggunakan algoritma CBC-mode dan RC5 dengan IV dan *key* yang telah di inisialisasi sebelumnya					
Keterangan:					

4.1.2.2 Mekanisme Pengiriman Data *Node* Pengirim dengan *Node Gateway*

REQ-ProTiny-sendData-2000 – Inisialisasi jalur radio komunikasi 2 *node*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `radio.openWritingPipe()` dan `radio.openReadingPipe()` berfungsi untuk melakukan inisialisasi jalur mana yang akan digunakan dalam proses komunikasi pengirm dan *node gateway*					
Keterangan:					

REQ-ProTiny-sendData-2001 – Pengiriman data dari *node* pengirim					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `radio.write()` berfungsi untuk mengirim data yang disimpan pada *payload* data kepada *node gateway*					
Keterangan:					

REQ-ProTiny-sendData-2002 – Penerimaan data oleh *node gateway*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `radio.read()` berfungsi untuk menerima data *payload* yang dikirim oleh *node* pengirim					
Keterangan:					

4.1.2.3 Mekanisme Dekripsi Data Protokol Keamanan TinySec *Node Gateway*

REQ-ProTiny-Decrypt-3000 – Penerimaan data *Ciphertext*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini berfungsi untuk menerima data dan menyimpan data pada sebuah variable yang telah dikirimkan oleh *node* pengirim					
Keterangan:					

REQ-ProTiny-Decrypt-3001 – Insiasi *Initialization Vector*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini berfungsi untuk mendapatkan *Initialization Vector* (IV) yang sama dengan IV pada proses enkripsi digunakan pada algoritma CBC-mode					
Keterangan:					

REQ-ProTiny-Decrypt-3002 – Inisialisasi *key* dan ekspansi *key*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini berfungsi untuk mendapatkan total 14 *key* yang sama pada proses enkripsi dengan panjang 16 bit tiap *key* sesuai dengan algoritma RC5					
Keterangan:					

REQ-ProTiny-Decrypt-3003 – Dekripsi data					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `Dekripsi()` berfungsi untuk melakukan enkripsi data yang telah diakuisisi, menggunakan algoritma CBC-mode dan RC5 dengan IV dan *key* yang telah di inisialisasi sebelumnya					
Keterangan:					

4.1.2.4 Mekanisme *Listening* Data *Ciphertext* pada *Node attacker*

REQ-ProTiny-*listen*Atck-4000 – Inisialisasi jalur radio komunikasi 2 *node*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `radio.openWritingPipe()` dan `radio.openReadingPipe()` berfungsi untuk melakukan inisialisasi jalur mana yang akan digunakan dalam proses komunikasi antara *node* pengirim dan *node attacker*					
Keterangan:					

REQ-ProTiny- *listen*Atck-4001 – Pengiriman data dari *node attacker*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `radio.write()` berfungsi untuk mengirm data yang disimpan pada *payload* data kepada *node attacker*					
Keterangan:					

REQ-ProTiny- *listen*Atck-4002 – Penerimaan data oleh *node gateway*					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `radio.read()` berfungsi untuk menerima data *payload* yang dikirim oleh *node* pengirim					

Keterangan:	

REQ-ProTiny- *listen*Atck-4003 – Pembacaan data					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Inspeksi
Fungsi perangkat lunak ini dinamakan `serial.print()` menampilkan data yang tidak dapat dibaca oleh *node attacker*					
Keterangan:					

4.2 Kebutuhan Non Fungsional

Kebutuhan Kebutuhan non-fungsional merupakan kebutuhan yang bersifat opsional agar sistem dapat berjalan lebih baik, namun apabila tidak dijalankan sistem tetep berjalan sesuai dengan fungsionalnya. Kode modul untuk kebutuhunan non fungsional adalah NF serta pemberian nomor kode dokumen kebutuhan non fungsional yakni 2000 sampai dengan 2001

4.2.1 Kebutuhan Peningkatan Performa *Key* dan IV

Pada bagian ini menjelaskan terkait dengan kebutuhan peningkatan performa penggunaan *key* dan *Initialization Vector* (IV). Bagian ini meliputi proses *update key* dan IV secara berkala pada *node* pengirim dan *node gateway*.

REQ-ProTiny- NF-2000 – *Update Key* dan IV					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Verifikasi:**	Demonstrasi
Update key berfungsi untuk melakukan *update key* pada *node* pengirim dan *node gateway*, dimana proses dilakukan secara berkala dengan cara melakukan *random key* dan IV dengan batasan sesuai jumlah bit yang dapat digunakan					
Keterangan:					

BAB 5 PERANCANGAN & IMPLEMENTASI

Pada bab ini memberikan penjelasan mengenai perancangan dan proses implementasi sistem yang meliputi perancangan sistem menggunakan diagram blok perangkat keras, skematik perangkat keras kemudian beberapa diagram alir yang ditanam pada perangkat sensor, aktuator maupun *gateway*. Implementasi sistem terdiri atas implementasi perangkat keras serta perangkat lunak.

5.1 Perancangan Sistem

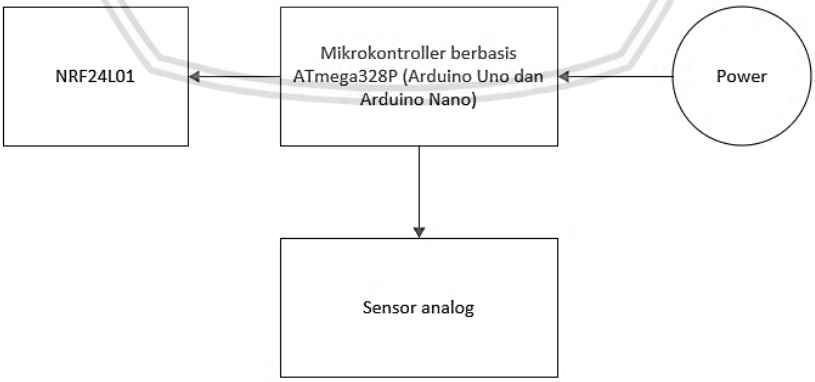
Pada bagian ini dijelaskan proses perancangan sistem yang dimulai dari perancangan perangkat keras meliputi diagram blok serta skematik perangkat keras dan perancangan perangkat lunak berupa diagram alir yang dijelaskan menggunakan *flow chart*.

5.1.1 Perancangan Perangkat Keras

Pada perancangan perangkat keras akan menghasilkan 2 metode, yaitu menggunakan diagram blok dan skematik perangkat keras.

5.1.1.1 Diagram Blok

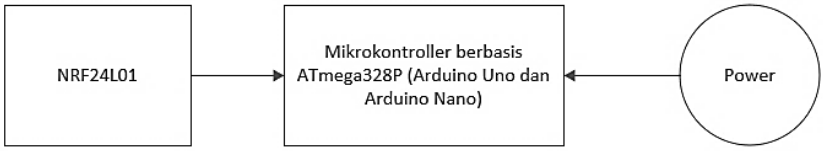
Pada sistem ini menerapkan 3 sistem perangkat keras meliputi perangkat keras *node* pengirim, *node gateway*, dan *node attacker*. Ketiga perangkat tersebut memiliki fungsionalitas yang berbeda, perangkat keras *node* pengirim memiliki fungsionalitas untuk melakukan akuisisi data dengan sensor serta melakukan proses enkripsi dan mengirimkan data hasil enkripsi berupa data *ciphertext* secara *node*. Perangkat keras *node gateway* memiliki fungsionalitas untuk menerima data *ciphertext* dari *node* pengirim, seingga dapat melakukan proses dekripsi data dan mendapatkan kembali *plaintext* data berupa nilai sensor yang telah di akuisisi oleh *node* pengirim. Perangkat keras *node attacker* memiliki fungsi *node* data yang dirimkan oleh *node* pengirim secara *node*, namun *node attacker* tidak dapat melakukan proses dekripsi.



Gambar 5.1 Diagram blok *node* pengirim

Peran yang dimiliki perangkat *node* pengirim sebagai perangkat yang dapat melakukan akuisisi data, proses enkripsi data, serta mengirimkan data hanya membutuh kan satu perangkat saja. Pada gambar 5.1, Mikrokontroler yang

digunakan pada *node* pengirim menggunakan mikrokontroler berbasis ATmega328p yaitu arduino uno atau arduino nano, perangkat keras sensor yang digunakan berbasis analog agar data yang didapatkan tidak bersifat diskrit salah satu sensor yang digunakan adala sensor MQ-2, dalam proses pengiriman data pernagkat *node* pengirim menggunakan NRF24L01 yang menggunakan frekuensi radio 2,4 Ghz yang dapat digunakan untuk melakukan komunikasi *wireless* pada sistem.

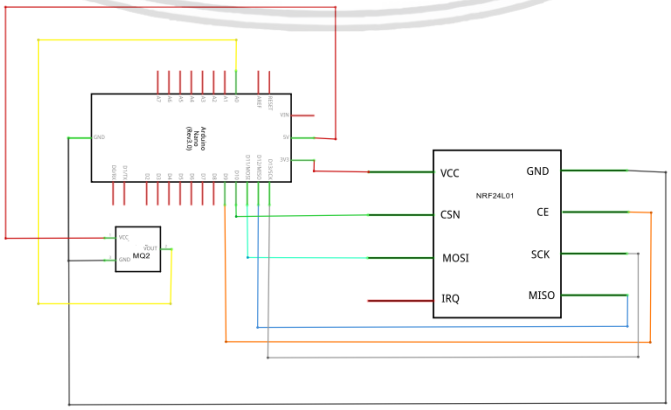


Gambar 5.2 Diagram blok *node gateway* dan *node attacker*

Peran yang dimiliki perangkat *node gateway* dan *node attacker* secara komunikasi memiliki peran yang sama, untuk menerima data dari *node* pengirim, sehingga pada gambar 5.2 *node gateway* dan *node attacker* memiliki diagram blok yang sama. Kedua perangkat memiliki mikrokontroler dengan basis yang sama, yaitu ATmega328p yang akan, namun berbeda dengan *node attacker* fungsi mikrokontroler pada *node gateway* memilik fungsi untuk melakukan dekripsi data. pada kedua perangkat menggunakan media komunikasi yang sama, yaitu NRF24l01 menyesuaikan dengan peprangkat *node* pengirim yang juga menggunakan perangkkat yang sama dengan komunikasi radio frekuensi. Seluruh perangkat membutukan sumber daya yang digunakan pada perangkat tersebut berasal dari *powerbank* atau pun melalu perangkat usb pada laptop dan terhubung mengunakan *micro-usb* pada perangkat mikrokontroler.

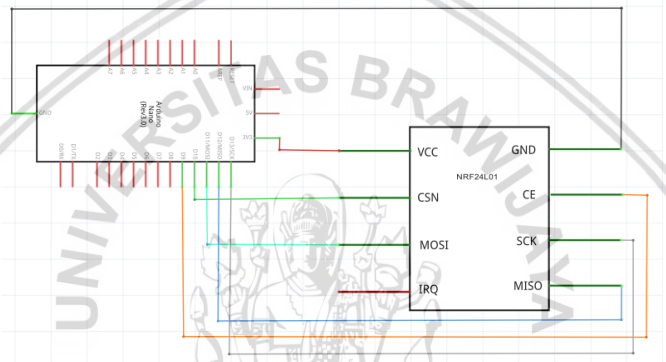
5.1.1.2 Skematik

Tahap perancangan berupa skematik ini penulis menggunakan perangkat lunak bernama Fritzung. Skematif memiliki fungsi sebagai pendekatan pada seluruh perangkat keras baik pada *node* pengirim, *node gateway*, dan *node attacker* terkait hubungan antara sumber daya, sensor, mikrokontroler, serta media komunikasi.



Gambar 5.3 Konfigurasi *node* pengirim

Pada gambar 5.3 digambarkan bahwa sensor MQ-2 terdiri dari 3 pin, yakni 1 pin yang dibutuhkan sebagai masukan tegangan yang di hubungkan pada pin 5v yang ada pada mikrokontroler. 1 pin lainnya terhubung dengan *ground* pada mikrokontroler dan 1 pin untuk Vout untuk mengirimkan data analog yang telah di akusisi dan terhubung pada pin analog (A0) pada mikrokontroler. Untuk terhubung dengan modul komunikasi NRF24L01 dengan total 7 pin yang terhubung dengan mikrontroler, yaitu pin GND yang terhubung dengan pin *ground* pada mikrontroller, pin VCC terhubung dengan pin 3.3v pada mikrontroler, pin CE terhubung dengan pin digital (D9) pada mikrokontroler, pin CSN terhubung dengan pin digital (D10) pada mikrokontroler, pin SCK terhubung dengan pin digital (D13) pada mikrokontroler, pin MOSI terhubung dengan pin digital (D11) pada mikrokontroler, pin MISO terhubung dengan pin digital (D12) pada mikrokontroler, penggunaan pin MISO, MOSI, dan SCK termasuk dalam penerapan komunikasi menggunakan *Serial Peripheral Interface* (SPI).



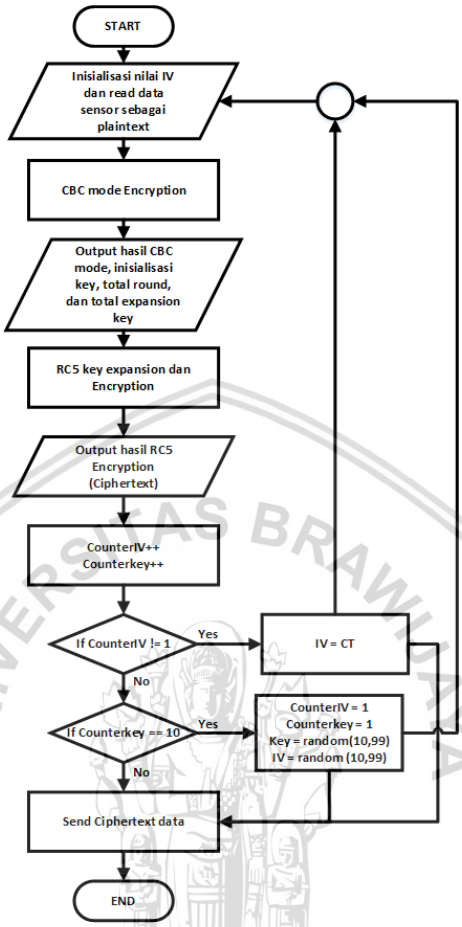
Gambar 5.4 Konfigurasi *node gateway* dan *node attacker*

Pada gambar 5.4 digambarkan bahwa berbeda dengan *node* pengirim, *node gateway* dan *node attacker* hanya menggunakan modul komunikasi NRF24L01 karena memiliki fungsionalitas untuk menerima data dari *node* pengirim, untuk penggunaan modul komunikasi NRF24L01 sama seperti konfigurasi pin pada *node* pengirim pin VCC terhubung dengan pin 3.3v pada mikrontroler, pin CE terhubung dengan pin digital (D9) pada mikrokontroler, pin CSN terhubung dengan pin digital (D10) pada mikrokontroler, pin SCK terhubung dengan pin digital (D13) pada mikrokontroler, pin MOSI terhubung dengan pin digital (D11) pada mikrokontroler, pin MISO terhubung dengan pin digital (D12) pada mikrokontroler.

5.1.2 Perancangan Perangkat Lunak

Pada perancangan perangkat lunak akan digambarkan kedalam bentuk diagram alir pada tiap proses yang terjadi tiap perangkat lunak yang ada pada node pengirim, *gateway*, dan *attacker*.

5.1.2.1 Diagram Alir *Node* Pengirim



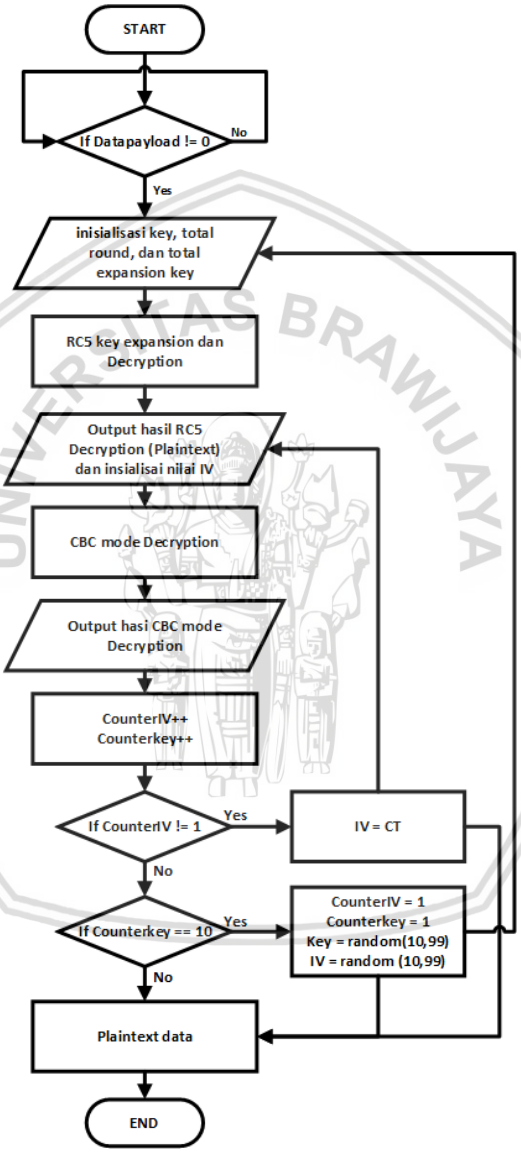
Gambar 5.5 Diagram alir *node* pengirim

Diagram alir pada proses sistem yang ada pada *node* pengirim di gambarkan dengan menggunakan flowchart sehingga dapat memberikan gambaran umum sistem pada sisi perangkat lunak dalam *node* pengirim. Pada *node* pengirim terjadi beberapa proses utama seperti proses penerapan CBC mode, RC5, dan pengiriman data. Pada gambar 5.5 jika digambarkan dalam betuk flowcart proses dimulai saat mikrokontroler dalam kondisi *start*, dimana proses pertama adalah melakukan inisialisasi nilai IV dan melakukan pembacaan akuisisi data sensor sebagai *plaintext*. Selanjutnya adalah proses CBC mode enkripsi dimana dalam proses tersebut dilakukan XOR antara nilai IV dengan nilai sensor.

Proses CBC mode akan menghasilkan *output* yang akan menjadi parameter *plaintext* untuk proses enkripsi menggunakan algoritma RC5, secara bersamaan juga menginisialisasinilai *key*, total *round*, dan total *expansion key*. Setelah menetukan beberapa parameter yang dibutuhkan proses enkripsi RC5, selanjutnya proses enkripsi akan di mulai dengan proses melakukan ekspansi *key* dan enkripsi, sehingga menghasilkan *output* berupa *ciphertext*, setelah proses enkripsi selesai selanjutnya akan ada parameter untuk melakukan proses *update*

key dan IV secara berkala sehingga dapat meningkatkan performa keamanan pada *key* dan IV dengan ketentuan menggunakan *counter*, selama *counter* atau proses sama dengan 10 maka nilai *key* dan IV akan di *random* dengan nilai min *random* = 10 dan maksimal = 99 karena sesuai dengan kebutuhan total 16 bits *key*. Selanjutnya proses terakhir yang terjadi pada *node* pengirim adalah melakukan pengiriman data secara *node*.

5.1.2.2 Diagram *Node Gateway*

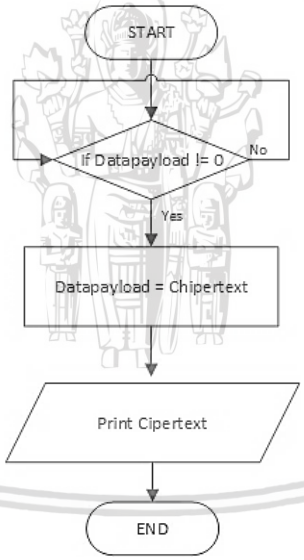


Gambar 5.6 Diagram alir *node gateway*

Diagram alir pada proses sistem yang ada pada *node gateway* digambarkan dengan menggunakan flowchart sehingga dapat memberikan gambaran umum sistem pada sisi perangkat lunak dalam *node gateway*. Pada gambar 5.6 jika digambarkan kedalam bentuk flowchart proses dimulai saat mikrokontroler dalam kondisi *start*, dimana proses pertama adalah menunggu data pada *payload* terisis,

hal ini terjadi ketika komunikasi melalui radio frekuensi berjalan, dan data yang di baca atau di read akan mengasilkan pada data *payload*, jika data *payload* telah didapatkan dan di anggap sebagai cipertext maka selanjutnya adalah proses insialisasi *key*, total *round*, dan total *expansion key*, yang dibutuhkan pada algoritma RC5, proses yang dilakukan adalah proses dekripsi data sehingga hasi dari proses RC5 adalah *plaintext*, selanjutnya untuk mendapatkan data *plaintext* yang sesuai dengan nilai data sensor yang didapatkan oleh *node* pengirim, proses berikutnya adalah melakukan CBC mode dekripsi, yaitu melakukan kembali XOR antara nilai IV dengan *plaintext* setelah proses RC5, setelah dialkukan proses CBC mode dekripsi akan menghasilkan *output* berupa *plaintext* yang telah sesuai dengan data sensor pada *node* pengirim. setelah proses dekripsi selesai selanjutnya akan ada parameter untuk melakukan proses *update key* dan IV secara berkala sehingga dapat meningkatkan performa keamanan pada *key* dan IV dengan ketentuan menggunakan *counter*, selama *counter* atau proses sama dengan 10 maka nilai *key* dan IV akan di *random* dengan nilai min *random* = 10 dan maksimal = 99 karena sesuai dengan kebutuhan total 16 bits *key*. Setelah mendapatkan nilai *plaintext* maka proses telah selesai.

5.1.2.3 Diagram Alir *node attacker*



Gambar 5.7 Diagram alir *node attacker*

Diagram alir pada proses sistem yang ada pada *node attacker* di gambarkan dengan mengunakan *flow chart* pada gambar 5.7 sehingga dapat memberikan gambaran umum sistem pada perangkat lunak dalam *node attacker*. Pada *node attacker* hanya memiliki 1 proses utama yaitu melakukan penerimaan data yang dikirimkan oleh *node* pengirim sehingga sebagai *node attacker* mendapatkan data *ciphertext* yang dikirimkan kepada *gateway*.

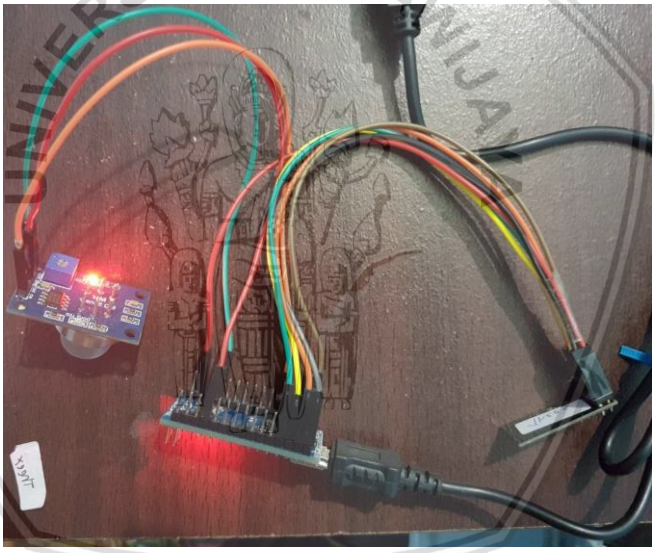
5.2 Implementasi Sistem

Pada bagian ini dijelaskan proses implementasi sistem yang dimulai dari implementasi perangkat keras yang terdiri dari *node* pengirim, *node gateway*, serta *node attacker*, dan implementasi perangkat lunak yang meliputi proses enkripsi data, dekripsi data, pengiriman data, *node* data dari *node attacker*, *update key* dan IV.

5.2.1 Impelementasi Perangkat Keras

5.2.1.1 Implementasi *node* pengirim

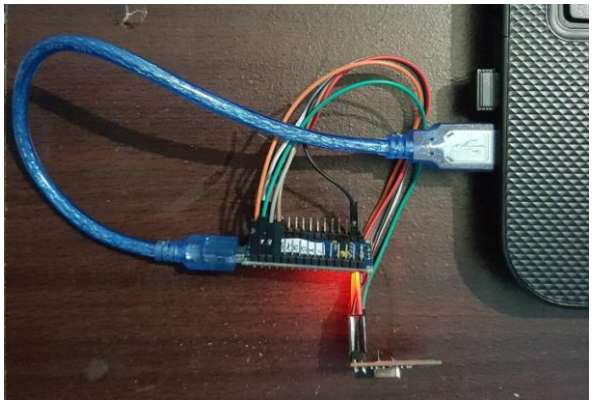
Sesuai dengan perancangan perangkat *node* pengirim, implementasi *node* pengirim terdiri atas mikrokontroler berbasis ATmega328p untuk sistem ini menggunakan Arduino nano sebagai perangkat pemroses data. sensor MQ2 sebagai perangkat yang melakukan akuisisi data analog yang mampu mendeteksi tingkat asap pada udara sekitar, serta dilengkapi dengan media komunikasi berupa perangkat NRF24L01 yang melakukan komunikasi *wireless* melalui radio frekuensi. Konfigurasi pin pada seluruh perangkat sesuai dengan perancangan pada gambar 5.3 dan masing-masing komponen terhubung dengan kabel *jumper*. Pada gambar 5.8 menunjukan hasil implementasi dari perancangan *node* pengirim.



Gambar 5.8 Perangkat *node* pengirim setelah dirakit

5.2.1.2 Implementasi *node gateway*

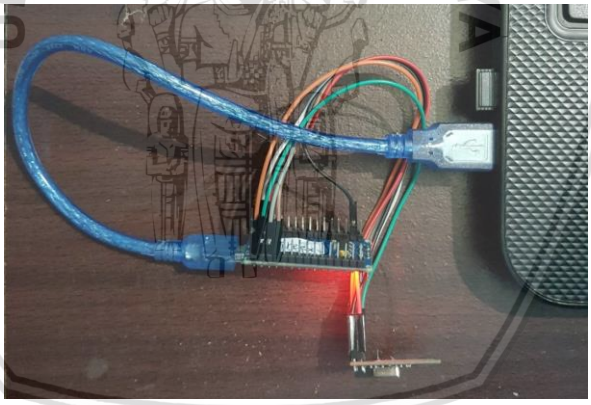
Sesuai dengan perancangan perangkat *node gateway*, implementasi *node gateway* terdiri atas mikrokontroler berbasis ATmega328p untuk sistem ini menggunakan Arduino nano sebagai perangkat pemroses data dan dilengkapi dengan media komunikasi berupa perangkat NRF24L01 yang melakukan komunikasi *wireless* melalui radio frekuensi. Konfigurasi pin pada seluruh perangkat sesuai dengan perancangan pada gambar 5.4 dan masing-masing komponen terhubung dengan kabel *jumper*. Pada gambar 5.9 menunjukan hasil implementasi dari perancangan *node gateway*.



Gambar 5.9 Perangkat *node gateway* setelah dirakit

5.2.1.3 Implementasi *node attacker*

Sesuai dengan perancangan perangkat *node attacker*, implementasi *node attacker* terdiri atas mikrokontroler berbasis ATmega328p untuk sistem ini menggunakan Arduino nano sebagai perangkat pemroses data dan dilengkapi dengan media komunikasi berupa perangkat NRF24L01 yang melakukan komunikasi *wireless* melalui radio frekuensi. Konfigurasi pin pada seluruh perangkat sesuai dengan perancangan pada gambar 5.4 dan masing-masing komponen terhubung dengan kabel *jumper*. Pada gambar 5.10 menunjukan hasil implementasi dari perancangan *node attacker*.



Gambar 5.10Perangkat *node attacker* setelah dirakit

5.2.2 Impelementasi Perangkat Lunak

5.2.2.1 Implementasi Mekanisme Enkripsi Data Protokol Keamanan TinySec *Node* Pengirim

Sesuai dengan rekayasa kebutuhan pada kode Encrypt dari nomor 1000 sampai dengan1003, akuisisi data sensor merupakan salah satu proses awal untuk mendapatkan data yang akan di enripsi. Implementasi kode program tersebut ada pada Tabel 5.1 yang diletakan pada fungsi loop pada program berbasiskan c++ namun memilki file ekstensi ino.

Tabel 5.1 Akuisisi data sensor

No	Kode Program
1	`Int data = analogRead(sensorpin);`
2	`Plaintext = fulldata0(String(data);`

Dalam proses enkripsi menggunakan dua algoritma yaitu RC5 dan CBC mode, implementasi kode program untuk melakukan enkripsi terdapat pada satu fungsi yaitu pada fungsi Enkripsi(), sebelum memasuki fungsi Enkripsi() inisialisasi beberapa parameter yang dibutukan dalam proses, yaitu *key* awal, IV awal, total sub*key*, panjang data, panjang *key*. Pada tabel 5.2 merupakan proses inisialisasi seluruh parameter secara global.

Tabel 5.2 Inisialisasi parameter

No	Kode Program
1	`String s[14];`
2	`String l[2];`
3	`String plaintext;`
4	`String masukan = "12";`
5	`String IV1 = "12", IV2 = "12";`
6	`String chipertext;`

Proses enkripsi algoritma RC5 juga memerlukan proses pembentukan sub *key* atau disebut juga dengan proses *key expansion* pada implementasi program terdapat pada Tabel 5.3 yang diletakkan pada fungsi loop.

Tabel 5.3 *Expansion key*

No	Kode Program
1	`String key = fullfill0(masukan);`
2	`s[0] = bin(dec("B7E1"));`
3	`String maks = bin(dec("9E37"));`
4	`for (int i = 1; i < 14; i++) {`
5	`String datas = Add(s[i - 1], maks);`
6	`s[i] = datas;}`
7	`for (int a = 0; a < 2; a++) {`
8	`int start = (1 - a) * 8;`
9	`int batas = ((1 - a) + 1) * 8;`
10	`String datal =`
11	`bin(dec(key.substring(start, batas)));`
12	`l[a] = datal;}`
13	`int i = 0, j = 0;`
14	`String c = "", d = "", tempr, tempr1,`
15	`tempr2, tempr3;`
16	`for (int count = 0; count < 42; count++) {`
17	`tempr1 = Add(fullfill0(c), fullfill0(d));`
18	`tempr = Add(s[i], tempr1);`
19	`c = tempr.substring(3) +`

No	Kode Program
20	tempr.substring(0, 3);
21	s[i] = c;
22	tempr2 = Add(fullfill0(c), fullfill0(d));
23	tempr3 = Add(l[j], tempr2);
24	String hexa = hex(tempr2);
25	long le = dec(hexa) % 16;
26	int len = le;
27	d = tempr3.substring(len) +
28	tempr3.substring(0, len);
29	l[j] = d;
30	i = (i + 1) % 14;
31	j = (j + 1) % 2;}

Setelah proses *expansion key* dilakukan proses enkripsi baru dapat dilakukan dengan memanggil fungsi enkripsi() pada fungsi loop. Implementasi fungsi enkripsi terdapat pada tabel 5.4. Fungsi enkripsi() memuat kedua algoritma secara langsung yaitu RC5 dan CBC mode, algoritma RC5 akan melakukan total *round* sesuai dengan yang telah diinisialisasi sebelumnya, dengan total panjang data *plaintext*, *key*, dan total jumlah sub*key* yang telah di ekspansi sebelumnya.

Tabel 5.4 Proses enkripsi RC5 dan CBC mode

No	Kode Program
1	String enkripsi () {
2	int panjang, x, y, tmp;
3	String pt1, pt2, finalpt;
4	Serial.print("Plaintext : ");
5	Serial.println(plaintext);
6	panjang = plaintext.length();
7	x = (panjang + 1) / 2;
8	y = panjang + 1;
9	pt1 = fullfill0(plaintext.substring(0, x));
10	pt2 = fullfill0(plaintext.substring(x, y));
11	pt1 = fullfill0(bin(dec(pt1)));
12	pt1 = eksor(pt1, IV1);
13	pt2 = fullfill0(bin(dec(pt2)));
14	pt2 = eksor(pt2, IV2);
15	pt1 = Add(pt1, s[0]);
16	pt2 = Add(pt2, s[1]);
17	for (int i = 1; i <= 6; i++) {
18	tmp = dec(hex(pt2)) % 16;
19	pt1 = eksor(pt1, pt2);
20	pt1 = pt1.substring(tmp) + pt1.substring(0,
21	tmp);
22	pt1 = Add(pt1, s[2 * i]);
23	tmp = dec(hex(pt1)) % 16;
24	pt2 = eksor(pt2, pt1);

No	Kode Program
25	pt2 = pt2.substring(tmp) + pt2.substring(0,
26	tmp);
27	pt2 = Add(pt2, s[(2 * i) + 1]);}
28	IV1 = pt1;
29	IV2 = pt2;
30	finalpt = pt1 + pt2;
31	String hasilct = hex(finalpt);
32	return hasilct;}

Setela kesleuruhan proses enkripsi dilakukan yaitu pada kode Encrypt 1000 sampai dengan 1003 akan mengasilkan cipertext yang siap untuk dikirim kepada *node gateway*.

5.2.2.2 Implementasi Mekanisme Pengiriman Data *Node* Pengirim dengan *Node Gateway*

Sesuai dengan rekayasa kebutuhan pada kode sendData dari nomor 2000 sampai dengan 2002, inisialisasi jalur radio komunikasi 2 *node* antara *node* pengirim dan *node gateway*, dengan menggunakan fungsi *radio.openWritingPipe()* pada *node* pengirm dan *radio.openReadingPipe()* pada *node gateway*, pada tabel 5.5 merupakan implementasi kode program inisialisasi jalur pada *node* pengirim.

Tabel 5.5 Inisialisasi jalur *node* pengirim

No	Kode Program
1	byte address[11] = "Simple*Node*";
2	radio.begin(); // Start up the radio
3	radio.openWritingPipe(address);

Jalur penerimaan data yang dilakukan pada ndoe *gateway*, dapat menggunakan fungsi *radio.openReadingPipe()*. Pada tabel 5.6 merupakan implementasi kode program inisialisasilisasi jalur pada *node gateway*.

Tabel 5.6 Inisialisasi jalur *node gateway*

No	Kode Program
1	byte address[11] = "Simple*Node*";
2	radio.begin(); // Start up the radio
3	radio.openReadingPipe(1, address);

Selanjutnya adalah proses pengiriman data pada *node* pengirim, dimana dalam prosesnya menggunakan fungsi radio.write() berfungsi untuk mengirm data yang disimpan pada *payload*. Pada tabel 5.7 merupakan implementasi kode program pengiriman data *payload*.

Tabel 5.7 Inisialisasi proses pengiriman data

No	Kode Program
1	radio.write(&payload, sizeof(unsigned long));

Setelah proses pengiriman selesai dilakukan, proses penerimaan data oleh *node gateway* dapat dimulai, dengan menggunakan fungsi *radio.read() node gateway* dapat menerima data *payload* yang telah dikirim sebelumnya oleh *node* pengirim. Pada tabel 5.8 merupakan implementasi kode program penerimaan data *payload*.

Tabel 5.8 Inisialisasi proses penerimaan data

No	Kode Program
1	`radio.read( &payload, sizeof(unsigned long) );`

Setelah data dapat diterima oleh *gateway*, berupa data *payload* makan selanjutnya adalah melakukan proses dekripsi data *payload* berupa *ciphertext* pada *node gateway*.

5.2.2.3 Implementasi Mekanisme Dekripsi Data Protokol Keamanan TinySec *Node Gateway*

Sesuai dengan rekayasa kebutuhan pada kode Decrypt dari nomor 3000 sampai dengan 3003, menerima data yang dikirim oleh *node* pengirim merupakan salah satu proses awal untuk mendapatkan data yang akan didekripsi dengan menggunakna fungsi *radio.read()* dan menyimpan daya *payload* pada sebuah variable, implementasi kode program tersebut ada pada Tabel 5.9.

Tabel 5.9 Inisialisasi jalur *node gateway*

No	Kode Program
1	`radio.read( &payload, sizeof(unsigned long) );`
2	`paystring = bin32(payload);`

Dalam proses dekripsi juga tetap menggunakan dua algoritma yaitu RC5 dan CBC mode, implementasi kode program untuk melakukan dekripsi terdapat pada satu fungsi yaitu pada fungsi dekripsi, sebelum memasuki fungsi dekripsi() inisialisasi beberapa parameter yang dibutukan dalam proses, yaitu *key* awal, IV awal, total sub*key*, panjang data, panjang *key*. Pada tabel 5.10 merupakan proses inisialisasi seluruh parameter secara global.

Tabel 5.10 Inisialisasi parameter

No	Kode Program
1	`String s[14];`
2	`String l[2];`
3	`String plaintext;`
4	`String masukan = "12";`
5	`String chipertext;`
6	`String paystring;`
7	`String IV1 = "12", IV2 = "12";`
8	`String IV3, IV4;`

Proses dekripsi algoritma RC5 juga memerlukan proses pembentukan sub *key* atau disebut juga dengan proses *key expansion* pada implementasi program terdapat pada Tabel 5.11 yang diletakkan pada fungsi loop.

Tabel 5.11 *Expansion key*

No	Kode Program
1	`String key = fullfill0(masukan);`
2	`    s[0] = bin(dec("B7E1"));`
3	`    String maks = bin(dec("9E37"));`
4	`    for (int i = 1; i < 14; i++) {`
5	`      String datas = Add(s[i - 1], maks);`
6	`      s[i] = datas;}`
7	`    for (int a = 0; a < 2; a++) {`
8	`      int start = (1 - a) * 8;`
9	`      int batas = ((1 - a) + 1) * 8;`
10	`      String datal =`
11	`bin(dec(key.substring(start, batas)));`
12	`      l[a] = datal;}`
13	`    int i = 0, j = 0;`
14	`    String c = "", d = "", tempr, tempr1,`
15	`tempr2, tempr3;`
16	`    for (int count = 0; count < 14; count++) {`
17	`      tempr1 = Add(fullfill0(c), fullfill0(d));`
18	`      tempr = Add(s[i], tempr1);`
19	`      c = tempr.substring(3) +`
20	`tempr.substring(0, 3);`
21	`      s[i] = c;`
22	`      tempr2 = Add(fullfill0(c), fullfill0(d));`
23	`      tempr3 = Add(l[j], tempr2);`
24	`      String hexa = hex(tempr2);`
25	`      long le = dec(hexa) % 16;`
26	`      int len = le;`
27	`      d = tempr3.substring(len) +`
28	`tempr3.substring(0, len);`
29	`      l[j] = d;`
30	`      i = (i + 1) % 14;`
31	`      j = (j + 1) % 2;}`

Setelah proses *expansion key* dilakukan proses dekripsi baru dapat dilakukan dengan memanggil fungsi dekripsi() pada fungsi loop. Implementasi fungsi enkripsi terdapat pada tabel 5.12. Fungsi dekripsi() memuat kedua algoritma secara langsung yaitu RC5 dan CBC mode, algoritma RC5 akan melakukan total *round* sesuai dengan yang telah diinisialisasi sebelumnya, dengan total panjang data *plaintext*, *key*, dan total jumlah sub*key* yang telah di ekspansi sebelumnya.

Tabel 5.12 Proses dekripsi RC5 dan CBC mode

No	Kode Program
1	`String dekripsi () {`
2	`  int tmp;`
3	`  String ct1, ct2, finalct, ctp1, ctp2;`

No	Kode Program
4	int panjang = paystring.length();
5	int x = (panjang + 1) / 2;
6	int y = panjang + 1;
7	ct1 = fullfill0(paystring.substring(0, x));
8	ct2 = fullfill0(paystring.substring(x, y));
9	ctp1 = (hex(ct1));
10	ctp2 = (hex(ct2));
11	Serial.print("Chyper text received : ");
12	Serial.println(ctp1 + ctp2);
13	ct1 = fullfill0(hex(ct1));
14	ct2 = fullfill0(hex(ct2));
15	ct1 = fullfill0(bin(dec(ct1)));
16	IV3 = ct1;
17	ct2 = fullfill0(bin(dec(ct2)));
18	IV4 = ct2;
19	for (int i = 6; i >= 1; i--) {
20	int pengurangan = (dec(hex(ct2))) -
21	(dec(hex(s[(2 * i) + 1])));
22	ct2 = fullfill0(bin(pengurangan));
23	tmp = dec(hex(ct1)) % 16;
24	ct2 = (ct2.substring(ct2.length() - tmp)) +
25	(ct2.substring(0, ct2.length() - tmp));
26	ct2 = eksor(ct2, ct1);
27	int pengurangan1 = (dec(hex(ct1))) -
28	(dec(hex(s[2 * i])));
29	ct1 = fullfill0(bin(pengurangan1));
30	tmp = dec(hex(ct2)) % 16;
31	ct1 = (ct1.substring(ct1.length() - tmp)) +
32	(ct1.substring(0, ct1.length() - tmp));
33	ct1 = eksor(ct1, ct2);
34	}
35	int last1 = (dec(hex(ct1))) -
36	(dec(hex(s[0])));
37	ct1 = fullfill0(bin(last1));
38	ct1 = eksor(ct1, IV1);
39	int last2 = (dec(hex(ct2))) -
40	(dec(hex(s[1])));
41	ct2 = fullfill0(bin(last2));
42	ct2 = eksor(ct2, IV2);
43	IV1 = IV3;
44	IV2 = IV4;
45	String finalpt = ct1.substring(8, 16) +
46	ct2.substring(8, 16);
47	String hasilpt = hex(finalpt);
48	Serial.print("got plain text : ");

No	Kode Program
49	Serial.println(hasilpt);
50	return hasilpt;}

Setela keseluruhan proses dekripsi dilakukan yaitu pada kode Decrypt 3000 sampai dengan 3003 akan mengasilkan *plaintext* yang sesuai dengan nilai sensor pada *node* pengirim.

5.2.2.4 Implementasi Mekanisme *Listening* Data *Ciphertext* pada *Node Attacker*

Sesuai dengan rekayasa kebutuhan pada kode listenAtck dari nomor 4000 sampai dengan 4003, inisialisasi jalur radio komunikasi 2 *node* antara *node* pengirim dan *node gateway*, dengan menggunakan fungsi radio.openWritingPipe() pada *node* pengirm dan radio.openReadingPipe() pada *node gateway*, pada tabel 5.13 merupakan implementasi kode program inisialisasi jalur pada *node* pengirim.

Tabel 5.13 Inisialisasi jalur *node* pengirim

No	Kode Program
1	byte address[11] = "Simple*Node*";
2	byte addresss[11] = "Simple*Node*1";
3	radio.begin(); // Start up the radio
4	radio.openWritingPipe(address);
5	radio.openWritingPipe(addresss);

Berbeda dengan proses pengiriman sebelumnya, pengiriman dengan adanya *node attacker* membuat proses inisialisasi jalur pada *node* pengirim degan 2 jalur. Jalur penerimaan data yang dilakukan pada *node attacker*, dapat menggunakan fungsi radio.openReadingPipe(). Pada tabel 5.14 merupakan implementasi kode program inisialisasilisasi jalur pada *node gateway*.

Tabel 5.14 Inisialisasi jalur *node attacker*

No	Kode Program
1	byte addresss[11] = "Simple*Node*1";
2	radio.begin(); // Start up the radio
3	radio.openReadingPipe(2, addresss);
4	radio.start*Node*();

Selanjutnya adalah proses pengiriman data pada *node* pengirim, dimana dalam prosesnya menggunakan fungsi radio.write() berfungsi untuk mengirim data yang disimpan pada *payload*. Pada tabel 5.15 merupakan implementasi kode program pengiriman data *payload*.

Tabel 5.15 Inisialisasi proses pengiriman data

No	Kode Program
1	radio.write(&payload, sizeof(unsigned long));

Setelah proses pengiriman selesai dilakukan, proses penerimaan data oleh *node attacker* dapat dimulai, dengan menggunakan fungsi radio.read() *node attacker* dapat menerima data *payload* yang telah dikirim sebelumnya oleh *node* pengirim. Pada tabel 5.16 merupakan implementasi kode program penerimaan data *payload*.

Tabel 5.16 Inisialisasi jalur *node gateway*

No	Kode Program
1	`radio.read( &payload, sizeof(unsigned long) );`

Setelah data dapat diterima oleh *gateway*, berupa data *payload* maka selanjutnya adalah melakukan proses print data *payload* untuk memastikan ada data yang masuk pada *node attacker*, dengan menggunakan fungsi Serial.println(). Pada Tabel 5.17 melakukan proses print data *payload* yang ditempatkan pada fungsi loop.

Tabel 5.17 Print data *payload*

No	Kode Program
1	`if (payload != 0) {`
2	`  Serial.println(payload);`
3	`}`

5.2.2.5 Implementasi *Update Key* dan IV

Sesuai dengan rekayasa kebutuhan pada kode ProTiny dari nomor 2000 sampai dengan 2003, proses *update key* dan IV dilakukan secara berkala, pada implementasi kode program dengan menginsiasi dimana *update key* dan IV akan dilakukan pada saat pengiriman dan penerimaan data dengan kelipatan 10, pada prosesnya melibatkan *counter* untuk menghitung total proses pengiriman dan penerimaan data. Pada Tabel 5.18 implementasi kode program untuk melakukan*update key* secara berkala.

Tabel 5.18 *Update key* dan IV

No	Kode Program
1	`if (count`*`key`*` == 10) {`
2	`  `*`key`*`rand = `*`random`*`(10, 99);`
3	`  masukan = String(`*`key`*`rand);`
4	`  IV1 = String(`*`key`*`rand);`
5	`  IV2 = String(`*`key`*`rand);`
6	`  count`*`key`*` = 1;`
7	`  countIV = 1;`
8	`}`
9	`Serial.print(count`*`key`*`);`
10	`count`*`key`*`++;`

Dengan adanya mekanisme *update key* dan IV menghasilkan peningkatan keamanan data *plaintext* maupun nilai *key* yang digunakan

BAB 6 PENGUJIAN

Pada bab ini menjelaskan tentang pengujian sistem yang sebelumnya telah diimplementasikan. Tahap ini dilakukan untuk mengetahui apakah sistem berjalan sesuai dengan kebutuhan fungsional serta non-fungsional yang telah dirancang sebelumnya pada bab rekayasa kebutuhan.

6.1 Pengujian Kebutuhan Fungsional

Pengujian kebutuhan fungsional bersifat kualitatif baik terhadap perangkat keras dan perangkat lunak dalam proses implementasi sistem. Pengujian pada kebutuhan fungsional disesuaikan dengan kebutuhan fungsional pada rekayasa kebutuhan.

6.1.1 Pengujian Mekanisme Enkripsi Data Protokol Keamanan TinySec *Node* Pengirim

Pengujian mekanisme enkripsi data protokol keamanan TinySec *node* pengirim yang mengacu pada rekayasa kebutuhan serta implementasi yang telah dijelaskan pada bab sebelumnya. Tujuan dari pengujian ini adalah untuk melakukan verifikasi terhadap hasil implementasi yang telah dilakukan. Proses awal yang harus dipenuhi dalam sistem ini adalah dimana proses akuisisi data sensor dapat dilakukan serta proses enkripsi data dengan menggunakan protokol keamanan TinySec berbasis algoritma RC5 dan CBC mode. Pengujian mekanisme ini memastikkan proses awal berjalan dengan baik dan sesuai dengan rekayasa kebutuhan.

Gambar 6.1 Enkripsi data pada *node* pengirim telah dilakukan

Pada gambar 6.1 menggambarkan bahwa proses enkripsi telah dilakukan melalui proses akuisisi data sensor yang digunakan sebagai *plaintext*, inisialisasi *key*, IV, ekspansi *key*, dan menghasilkan *Ciphertext*.

Tabel 6.1 Tabel pengujian mekanisme enkripsi data protokol keamanan TinySec *node* pengirim

No.	Test Name	Test Case	Expected Result	Result	Status
1.	REQ-ProTiny-Encrypt-1000-Akuisisi data sensor	Melakukan pengujian pada fungsi `analogRead (pin)`	Perangkat sensor pada *node* pengirim dapat melakukan akuisisi data dan data disimpan sebagai *plaintext*	Perangkat sensor pada *node* pengirim dapat melakukan akuisisi data dan data dapat disimpan sebagai *plaintext*	Valid
2.	REQ-ProTiny-Encrypt-1001-Insialisasi Intialization Vector	Melakukan pengujian perangkat lunak memiliki fungsi untuk mendapatkan atau menginisialisasi Intialization Vector (IV)	Perangkat *node* pengirim dapat menginisialisasi nilai IV yang dibutuhkan sebagai parameter pada CBC mode	Perangkat *node* pengirim dapat menginisialisasi nilai IV yang dibutuhkan sebagai parameter pada CBC mode dengan nilai IV pertama ditentukan sejak awal	Valid
3.	REQ-ProTiny-Encrypt-1002-Inisialisasi *key* dan ekspansi *key*	Melakukan pengujian perangkat lunak memiliki fungsi untuk menginisialisasi *key* dan melakukan ekspansi *key*	Perangkat *node* pengirim dapat melakukan inisiasi *key* dan melakukan ekspansi *key*	Perangkat *node* pengirim dapat melakukan inisiasi *key* awal dan melakukan ekspansi	Valid

No.	Test Name	Test Case	Expected Result	Result	Status
			dengan total yang yang telah ditentukan	*key* hingga mengahasilkan 14 sub*key*	
4.	REQ-ProTiny-Encrypt-1003-Enkripsi data	Melakukan pengujian pada fungsi `enkripsi()` untuk melakukan enkripsi data yang telah di akusisi menggunakan algoritma CBC-mode dan RC5	Perangkat *node* pengirim dapat menghasilkan *ciphertext* dari proses enkripsi sehingga cipertext siap untuk dikirim ke *node gateway*	Perangkat *node* pengirim dapat mengasilkan *ciphertext* dari proses enkripsi dan *ciphertext* siap dikirim kepada *node gateway*	Valid

Dari hasil pengujian diatas, semua proses dalam mekanisme enkrpsi data protokol keamanan *node* pengirim dinyatakan valid dan berhasil dilakukan. Jika di analisa proses mekanisme enkripsi protokol keamanan TinySec berbasis algoritma RC5 dan CBC mode enkripsi tersebut dapat dijelaskan sebagai berikut.

1. Proses Inisialisasi parameter yang dibutuhkan dalam proses enkripsi

   Key = 12 = 0000 0000 0001 0010

   Pw = B7E1 = 1011 0111 1110 0001

   Qw = 9E37 = 1001 1110 0011 0111

   Data = 0037 = 0000 0000 0011 0111

   r = 6

   t = 2 (r+1) = 14

   b = 2

2. Langkah selanjutnya melakukan *key expansion*, jika sesuai dengan dasar teori yang telah dijelaskan sebelumnya mengenai proses enkripsi algoritma RC5, proses *key expansion* memiliki 3 proses utama proses merubah key rahasia dari bytes menjadi *word*s, menginisiasi seluruh nilai dari array S, dan proses *mixing secret key*.

2.1. Proses merubah key rahasia dari *word*s menjadi bytes, proses yang dilakukan

Key = 12 = 0000 0000 0001 0010

For i = b-1 downto 0 do

L[i/u] = (L[i/u] <<< 8) + K[i]

Proses perulangan yang berjalan sebagai berikut

i = 1

L[0] = 0000 0000 0001 0010

i = 0

L[1] = 0000 0000 0000 0000

2.2. Proses selanjutnya adalah melakukan inisialisasi nilai dari array S, yang akan menyimpan nilai key yang telah di ekspansi

S[0] = Pw = 1011 0111 1110 0001
Insialisasi sub key sejumlah t = 14 dengan menggunkana algoritma sebagai berikut
For (int i = 1; i<t; i++){
s[i] = s[i-1] + Qw}
Proses perulangan yang dilakukan sesuai algoritma di atas sebagai berikut
i = 1
s[1] = s[0] + Qw
= 1011 0111 1110 0001 + 1001 1110 0011 0111
= 1 0101 0110 0001 1000
i = 2
s[2] = s[1] + Qw
= 1 0101 0110 0001 1000 + 1001 1110 0011 0111
= 1 1111 0100 0100 1111
i = 3
s[3] = s[2] + Qw
= 1 1111 0100 0100 1111 + 1001 1110 0011 0111
= 10 1001 0010 1000 0110
i = 4
s[4] = s[3] + Qw
= 10 1001 0010 1000 0110+ 1001 1110 0011 0111
= 11 0011 0000 1011 1101
i = 5
s[5] = s[4] + Qw
= 11 0011 0000 1011 1101 + 1001 1110 0011 0111
= 11 1100 1110 1111 0100

i = 6
s[6] = s[5] + Qw
= 11 1100 1110 1111 0100 + 1001 1110 0011 0111
= 100 0110 1101 0010 1011
i = 7
s[7] = s[6] + Qw
= 100 0110 1101 0010 1011 + 1001 1110 0011 0111
= 101 0000 1011 0110 0010
i = 8
s[8] = s[7] + Qw
= 101 0000 1011 0110 0010 + 1001 1110 0011 0111
= 101 1010 1001 1001 1001
i = 9
s[9] = s[8] + Qw
= 101 1010 1001 1001 1001 + 1001 1110 0011 0111
= 110 0100 0111 1101 0000
i = 10
s[10] = s[9] + Qw
= 110 0100 0111 1101 0000 + 1001 1110 0011 0111
= 110 1110 0110 0000 0111
i = 11
s[11] = s[10] + Qw
= 110 1110 0110 0000 0111 + 1001 1110 0011 0111
= 111 1000 0100 0011 1110
i = 12
s[12] = s[11] + Qw
= 111 1000 0100 0011 1110 + 1001 1110 0011 0111
= 1000 0010 0010 0111 1110
i = 13
s[13] = s[12] + Qw
= 1000 0010 0010 0111 1110 + 1001 1110 0011 0111
= 1000 1100 0000 1010 1100

2.3. Proses terakhir yang perlu dilakukan dalam proses *key expansion* melakukan proses *mixing secret key*.

i = j = 0

A = B = 0

do 42 times:

A = S[i] = (S[i] + A + B) <<< 3

B = L[j] = (L[j] + A + B) <<< ((A + B)%16)

i = (i + 1) % t

j = (j + 1) % c

dari algoritma di atas, maka perulangan yang dilakukan adalah sebagai berikut

do 1 :

A = S[0] = (S[0] + A + B) <<< 3

= 1011 0111 1110 0001 + 0000 0000 0000 0000 + 0000 0000 0000 0000

= 1011 0111 1110 0001 <<< 3

= 1011 1111 0000 1101

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0000 0000 0001 0010 + 1011 1111 0000 1101 + 0000 0000 0000 0000

= 1011 1111 0001 1111 <<< 13

= 1111 0111 1110 0011

i = (0 + 1) % 14 = 1

j = (0 + 1) % 2 = 1

do 2 :

A = S[1] = (S[1] + A + B) <<< 3

= 0101 0110 0001 1000 + 1011 1111 0000 1101 + 1111 0111 1110 0011

= 0000 1101 0000 1000 <<< 3

= 0110 1000 0100 0000

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0000 0000 0000 0000 + 0110 1000 0100 0000 + 1111 0111 1110 0011

= 0110 0000 0010 0011 <<< 3

= 0000 0001 0001 1011

i = (1 + 1) % 14 = 2

j = (1 + 1) % 2 = 0

do 3 :

A = S[2] = (S[2] + A + B) <<< 3

= 1111 0100 0100 1111 + 0110 1000 0100 0000 + 0000 0001 0001 1011

= 0101 1101 1010 1010 <<< 3

= 1110 1101 0101 0010

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 1111 0111 1110 0011 + 1110 1101 0101 0010 + 0000 0001 0001 1011

= 1110 0110 0101 0000 <<< 13

= 0001 1100 1100 1010

i = (2 + 1) % 14 = 3

j = (0 + 1) % 2 = 1

do 4:

A = S[3] = (S[3] + A + B) <<< 3

= 1001 0010 1000 0110 + 1110 1101 0101 0010 + 0001 1100 1100 1010

= 1001 1100 1010 0010 <<< 3

= 1110 0101 0001 0100

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0000 0001 0001 1011 + 1110 0101 0001 0100 + 0001 1100 1100 1010

= 0000 0010 1111 1001 <<< 14

= 0100 0000 1011 1110

i = (3 + 1) % 14 = 4

j = (1 + 1) % 2 = 0

do 5:

A = S[4] = (S[4] + A + B) <<< 3

= 0011 0000 1011 1101 + 1110 0101 0001 0100 + 0100 0000 1011 1110

= 0101 0110 1000 1111 <<< 3

= 1011 0100 0111 1010

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0001 1100 1100 1010 + 1011 0100 0111 1010 + 0100 0000 1011 1110

= 0001 0010 0000 0010 <<< 8

= 0000 0010 0001 0010

i = (4 + 1) % 14 = 5

j = (0 + 1) % 2 = 1

do 6:

A = S[5] = (S[5] + A + B) <<< 3

= 1100 1110 1111 0100 + 1011 0100 0111 1010 + 0000 0010 0001 0010

= 1000 0101 1000 0000 <<< 3

= 0010 1100 0000 0100

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0100 0000 1011 1110 + 0010 1100 0000 0100 + 0000 0010 0001 0010

= 0110 1110 1101 0100 <<< 6

= 1011 0101 0001 1011

i = (5 + 1) % 14 = 6

j = (1 + 1) % 2 = 0

do 7:

A = S[6] = (S[6] + A + B) <<< 3

= 0110 1101 0010 1011 + 0010 1100 0000 0100 + 1011 0101 0001 1011

= 0100 1110 0100 1010 <<< 3

= 0111 0010 0101 0010

B = L[0] = (L[1] + A + B) <<< ((A + B)%16)

= 0000 0010 0001 0010 + 0111 0010 0101 0010 + 1011 0101 0001 1011

= 0010 1001 0111 1111 <<< 13

= 1110 0101 0010 1111

i = (6 + 1) % 14 = 7

j = (0 + 1) % 2 = 1

do 8:

A = S[7] = (S[7] + A + B) <<< 3

= 0000 1011 0110 0010 + 0111 0010 0101 0010 + 1110 0101 0010 1111

= 0110 0010 1110 0011 <<< 3

= 0001 0111 0001 1011

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1011 0101 0001 1011 + 0001 0111 0001 1011 + 1110 0101 0010 1111

= 1011 0001 0110 0101 <<< 10

= 1001 0110 1100 0101

i = (7 + 1) % 14 = 8

j = (1 + 1) % 2 = 0

do 9:

A = S[8] = (S[8] + A + B) <<< 3

= 1010 1001 1001 1001 + 0001 0111 0001 1011 + 1001 0110 1100 0101

= 0101 0111 0111 1001 <<< 3

= 1011 1011 1100 1010 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 1110 0101 0010 1111 + 1011 1011 1100 1010 + 1001 0110 1100 0101

= 0011 0111 1011 1110<<< 15

= 0001 1011 1101 1111

i = (8 + 1) % 14 = 9

j = (0 + 1) % 2 = 1

do 10:

A = S[9] = (S[9] + A + B) <<< 3

= 0100 0111 1101 0000 + 1011 1011 1100 1010 + 0001 1011 1101 1111

= 0001 1111 0111 1001 <<< 3

= 1111 1011 1100 1000

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1001 0110 1100 0101 + 1011 1011 1100 1010 + 1001 0110 1100 0101

= 1010 1110 0110 1100 <<< 7

= 0011 0110 0101 0111

i = (9 + 1) % 14 = 10

j = (1 + 1) % 2 = 0

do 11:

A = S[10] = (S[10] + A + B) <<< 3

= 1110 0110 0000 0111 + 1111 1011 1100 1000 + 0011 0110 0101 0111

= 0001 1000 0010 0110 <<< 3

= 1100 0001 0011 0000 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0001 1011 1101 1111 + 1100 0001 0011 0000 + 0011 0110 0101 0111

= 0001 0011 0110 0110 <<< 7

= 1011 0011 0000 1001 i = (10 + 1) % 14 = 11

j = (0 + 1) % 2 = 1

do 12:

A = S[11] = (S[11] + A + B) <<< 3

= 1000 0100 0011 1110 + 1100 0001 0011 0000 + 1011 0011 0000 1001

= 1111 1000 0111 0111 <<< 3

= 1100 0011 1011 1111 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0011 0110 0101 0111 + 1100 0011 1011 1111 + 1011 0011 0000 1001

= 1010 1101 0001 1111 <<< 8

= 0001 1111 1010 1101

i = (11 + 1) % 14 = 12

j = (1 + 1) % 2 = 0

do 13:

A = S[12] = (S[12] + A + B) <<< 3

= 0010 0010 0111 0101 + 1100 0001 0011 0000 + 0001 1111 1010 1101

= 0000 0101 1110 0001 <<< 3

= 0010 1111 0000 1000 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 1011 0011 0000 1001 + 0010 1111 0000 1000 + 0001 1111 1010 1101

= 0000 0001 1011 1110 <<< 5

= 0011 0111 1100 0000

i = (12 + 1) % 14 = 13

j = (0 + 1) % 2 = 1

do 14:

A = S[13] = (S[13] + A + B) <<< 3

= 1100 0000 1010 1100 + 0010 1111 0000 1000 + 0011 0111 1100 0000

= 0010 0111 0111 0100 <<< 3

= 0011 1011 1010 0001 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0001 1111 1010 1101 + 0011 1011 1010 0001 + 0011 0111 1100 0000

= 1001 0011 0000 1110 <<< 1

= 0010 0110 0001 1101 i = (13 + 1) % 14 = 0

j = (1 + 1) % 2 = 0

do 15 :

A = S[0] = (S[0] + A + B) <<< 3

= 1011 1111 0000 1101 + 0011 1011 1010 0001 + 0010 0110 0001 1101

= 0010 0000 1100 1011 <<< 3

= 0000 0110 0101 1001

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0011 0111 1100 0000 + 0000 0110 0101 1001 + 0010 0110 0001 1101

= 0110 0100 0011 0110 <<< 6

= 0000 1101 1001 1001

i = (0 + 1) % 14 = 1

j = (0 + 1) % 2 = 1

do 16 :

A = S[1] = (S[1] + A + B) <<< 3

= 0110 1000 0100 0000 + 0000 0110 0101 1001 + 0000 1101 1001 1001

= 0111 1100 0011 0010 <<< 3

= 1110 0001 1001 0011

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0010 0110 0001 1101 + 1110 0001 1001 0011 + 0000 1101 1001 1001

= 0001 0101 0100 1001 <<< 12

= 1001 0001 0101 0100

i = (1 + 1) % 14 = 2

j = (1 + 1) % 2 = 0

do 17 :

A = S[2] = (S[2] + A + B) <<< 3

= 1110 1101 0101 0010 + 1110 0001 1001 0011 + 1001 0001 0101 0100

= 0110 0000 0011 1001 <<< 3

= 0000 0001 1100 1011

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0000 1101 1001 1001 + 0000 0001 1100 1011 + 1001 0001 0101 0100

= 1010 0000 1011 1000 <<< 15

= 0101 0000 0101 1100

i = (2 + 1) % 14 = 3

j = (0 + 1) % 2 = 1

do 18:

A = S[3] = (S[3] + A + B) <<< 3

= 1110 0101 0001 0100 + 0000 0001 1100 1011 + 0101 0000 0101 1100

= 0011 0111 0011 1011 <<< 3

= 1011 1001 1101 1001

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1001 0001 0101 0100 + 1011 1001 1101 1001 + 0101 0000 0101 1100

= 1001 1011 1000 1001 <<< 5

= 0111 0001 0011 0011

i = (3 + 1) % 14 = 4

j = (1 + 1) % 2 = 0

do 19:

A = S[4] = (S[4] + A + B) <<< 3

= 1011 0100 0111 1010 + 1011 1001 1101 1001 + 0111 0001 0011 0011

= 1101 1111 1000 0110 <<< 3

= 1111 1100 0011 0110

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0101 0000 0101 1100 + 1111 1100 0011 0110 + 0111 0001 0011 0011

= 1011 1101 1100 0101 <<< 9

= 1000 1011 0111 1011

i = (4 + 1) % 14 = 5

j = (0 + 1) % 2 = 1

do 20:

A = S[5] = (S[5] + A + B) <<< 3

= 0010 1100 0000 0100 + 1111 1100 0011 0110 + 1000 1011 0111 1011

= 1011 0011 1011 0101 <<< 3

= 1001 1101 1010 1101

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0111 0001 0011 0011 + 1111 1100 0011 0110 + 1000 1011 0111 1011

= 1001 1010 0101 1011 <<< 8

= 0101 1011 1001 1010

i = (5 + 1) % 14 = 6

j = (1 + 1) % 2 = 0

do 21:

A = S[6] = (S[6] + A + B) <<< 3

= 0111 0010 0101 0010 + 1001 1101 1010 1101 + 0101 1011 1001 1010

= 0110 1011 1001 1001 <<< 3

= 0101 1100 1100 1011

B = L[0] = (L[1] + A + B) <<< ((A + B)%16)

= 0000 0010 0001 0010 + 0101 1100 1100 1011 + 0101 1011 1001 1010

= 0100 0011 1110 0000 <<< 5

= 0111 1100 0000 1000

i = (6 + 1) % 14 = 7

j = (0 + 1) % 2 = 1

do 22:

A = S[7] = (S[7] + A + B) <<< 3

= 0001 0111 0001 1011 + 0101 1100 1100 1011 + 0111 1100 0000 1000

= 1110 1111 1110 1110 <<< 3

= 0111 1111 0111 0111 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0101 1011 1001 1010 + 0111 1111 0111 0111 + 0111 1100 0000 1000

= 0101 0111 0001 1001 <<< 15

= 1010 1011 1000 1100

i = (7 + 1) % 14 = 8

j = (1 + 1) % 2 = 0

do 23:

A = S[8] = (S[8] + A + B) <<< 3

= 1011 1011 1100 1010 + 0111 1111 0111 0111 + 1010 1011 1000 1100

= 1110 0110 1100 1101 <<< 3

= 0011 0110 0110 1111 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0111 1100 0000 1000 + 0011 0110 0110 1111 + 1010 1011 1000 1100

= 0101 1110 0000 0011 <<< 11

= 0001 1010 1111 0000

i = (8 + 1) % 14 = 9

j = (0 + 1) % 2 = 1

do 24:

A = S[9] = (S[9] + A + B) <<< 3

= 1111 1011 1100 1000 + 0011 0110 0110 1111 + 0001 1010 1111 0000

= 0100 1101 0010 0111 <<< 3

= 0110 1001 0011 1010

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1010 1011 1000 1100 + 0110 1001 0011 1010 + 0001 1010 1111 0000

= 0010 1111 1011 0110 <<< 10

= 1101 1000 1011 1110

i = (9 + 1) % 14 = 10

j = (1 + 1) % 2 = 0

do 25:

A = S[10] = (S[10] + A + B) <<< 3

= 1100 0001 0011 0000 + 0110 1001 0011 1010 + 1101 1000 1011 1110

= 0000 0011 0010 1000 <<< 3

= 0001 1001 0100 0000 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0001 1010 1111 0000 + 0001 1001 0100 0000 + 1101 1000 1011 1110

= 0000 1100 1110 1110 <<< 14

= 1000 0011 0011 1011 i = (10 + 1) % 14 = 11

j = (0 + 1) % 2 = 1

do 26:

A = S[11] = (S[11] + A + B) <<< 3

= 1100 0011 1011 1111 + 0001 1001 0100 0000 + 1000 0011 0011 1011

= 0110 0000 0011 1010 <<< 3

= 0000 0001 1101 0011 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1101 1000 1011 1110 + 0000 0001 1101 0011 + 1000 0011 0011 1011

= 0101 1101 1100 1100 <<< 14

= 0001 0111 0111 0011 i = (11 + 1) % 14 = 12

j = (1 + 1) % 2 = 0

do 27:

A = S[12] = (S[12] + A + B) <<< 3

= 0010 1111 0000 1000 + 0000 0001 1101 0011 + 0001 0111 0111 0011

= 0100 1000 0100 1110 <<< 3

= 0100 0010 0111 0010 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 1000 0011 0011 1011 + 0100 0010 0111 0010 + 0001 0111 0111 0011

= 1101 1101 0010 0000 <<< 5

= 1010 0100 0001 1011

i = (12 + 1) % 14 = 13

j = (0 + 1) % 2 = 1

do 28:

A = S[13] = (S[13] + A + B) <<< 3

= 0011 1011 1010 0001 + 0100 0010 0111 0010 + 1010 0100 0001 1011

= 0010 0010 0010 1110 <<< 3

= 0001 0001 0111 0001 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0001 0111 0111 0011 + 0001 0001 0111 0001 + 1010 0100 0001 1011

= 1100 1100 1111 1111 <<< 12

= 1111 1100 1100 1111 i = (13 + 1) % 14 = 0

j = (1 + 1) % 2 = 0

do 29 :

A = S[0] = (S[0] + A + B) <<< 3

= 0000 0110 0101 1001 + 0001 0001 0111 0001 + 1111 1100 1100 1111

= 0001 0100 1001 1001 <<< 3

= 1010 0100 1100 1000

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 1010 0100 0001 1011 + 1010 0100 1100 1000 + 1111 1100 1100 1111

= 0100 0101 1011 0010 <<< 7

= 1101 1001 0010 0010

i = (0 + 1) % 14 = 1

j = (0 + 1) % 2 = 1

do 30 :

A = S[1] = (S[1] + A + B) <<< 3

= 1110 0001 1001 0011 + 1010 0100 1100 1000 + 1101 1001 0010 0010

= 0101 1111 0111 1101 <<< 3

= 1111 1011 1110 1010 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1111 1100 1100 1111 + 1111 1011 1110 1010 + 1101 1001 0010 0010

= 1101 0001 1101 1011 <<< 12

= 1011 1101 0001 1101

i = (1 + 1) % 14 = 2

j = (1 + 1) % 2 = 0

do 31 :

A = S[2] = (S[2] + A + B) <<< 3

= 0000 0001 1100 1011 + 1111 1011 1110 1010 + 1011 1101 0001 1101

= 1011 1010 1101 0010 <<< 3

= 1101 0110 1001 0101

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 1101 1001 0010 0010 + 1101 0110 1001 0101 + 1011 1101 0001 1101

= 0110 1100 1101 0100 <<< 2

= 1011 0011 0101 0001

i = (2 + 1) % 14 = 3

j = (0 + 1) % 2 = 1

do 32:

A = S[3] = (S[3] + A + B) <<< 3

= 1011 1001 1101 1001 + 1101 0110 1001 0101 + 1011 0011 0101 0001

= 0100 0011 1011 1111 <<< 3

= 0001 1101 1111 1010

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1001 0001 0101 0100 + 0001 1101 1111 1010 + 1011 0011 0101 0001

= 1000 1110 0110 1000 <<< 11

= 0100 0100 0111 0011

i = (3 + 1) % 14 = 4

j = (1 + 1) % 2 = 0

do 33:

A = S[4] = (S[4] + A + B) <<< 3

= 1111 1100 0011 0110 + 0001 1101 1111 1010 + 0100 0100 0111 0011

= 0101 1110 1010 0011 <<< 3

= 1111 0101 0001 1010

B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 1011 0011 0101 0001 + 1111 0101 0001 1010 + 0100 0100 0111 0011

= 1110 1100 1101 1110 <<< 13

= 1101 1101 1001 1011

i = (4 + 1) % 14 = 5

j = (0 + 1) % 2 = 1

do 34:

A = S[5] = (S[5] + A + B) <<< 3

= 1001 1101 1010 1101 + 1111 0101 0001 1010 + 1101 1101 1001 1011

= 0111 0000 0110 0010 <<< 3

= 1000 0011 0001 0011

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0100 0100 0111 0011 + 1000 0011 0001 0011 + 1101 1101 1001 1011

= 1010 0101 0010 0001 <<< 14

= 0110 1001 0100 1000

i = (5 + 1) % 14 = 6

j = (1 + 1) % 2 = 0

do 35:

A = S[6] = (S[6] + A + B) <<< 3

= 0101 1100 1100 1011 + 1000 0011 0001 0011 + 0110 1001 0100 1000

= 0100 1001 0010 0110 <<< 3

= 0100 1001 0011 0010 B = L[0] = (L[1] + A + B) <<< ((A + B)%16)

= 1101 1101 1001 1011 + 0100 1001 0011 0010 + 0110 1001 0100 1000

= 1001 0000 0001 0101 <<< 10

= 0101 0110 0100 0000

i = (6 + 1) % 14 = 7

j = (0 + 1) % 2 = 1

do 36:

A = S[7] = (S[7] + A + B) <<< 3

= 0111 1111 0111 0111 + 0100 1001 0011 0010 + 0101 0110 0100 0000

= 0001 1110 1110 1001 <<< 3

= 1111 0111 0100 1000 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 0110 1001 0100 1000 + 1111 0111 0100 1000 + 0101 0110 0100 0000

= 1011 0110 1101 0000 <<< 8

= 1101 0000 1011 0110

i = (7 + 1) % 14 = 8

j = (1 + 1) % 2 = 0

do 37:

A = S[8] = (S[8] + A + B) <<< 3

= 0011 0110 0110 1111 + 1111 0111 0100 1000 + 1101 0000 1011 0110

= 1111 1110 0110 1101 <<< 3

= 1111 0011 0110 1111 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0101 0110 0100 0000 + 1111 0011 0110 1111 + 1101 0000 1011 0110

= 0001 1010 0110 0101 <<< 5

= 0100 1100 1010 0011

i = (8 + 1) % 14 = 9

j = (0 + 1) % 2 = 1

do 38:

A = S[9] = (S[9] + A + B) <<< 3

= 0110 1001 0011 1010 + 1111 0011 0110 1111 + 0100 1100 1010 0011

= 1010 1001 0100 1100 <<< 3

= 0100 1010 0110 0101

B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1101 0000 1011 0110 + 0100 1010 0110 0101 + 0100 1100 1010 0011

= 0110 0111 1011 1110 <<< 8

= 1011 1110 0110 0111

i = (9 + 1) % 14 = 10

j = (1 + 1) % 2 = 0

do 39:

A = S[10] = (S[10] + A + B) <<< 3

= 0001 1001 0100 0000 + 0100 1010 0110 0101 + 1011 1110 0110 0111

= 0010 0010 0000 1100 <<< 3

= 0001 0000 0110 0001 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0100 1100 1010 0011 + 0001 0000 0110 0001 + 1011 1110 0110 0111

= 0001 1011 0110 1011 <<< 8

= 0110 1011 0001 1011 i = (10 + 1) % 14 = 11

j = (0 + 1) % 2 = 1

do 40:

A = S[11] = (S[11] + A + B) <<< 3

= 0000 0001 1101 0011 + 0001 0000 0110 0001 + 0110 1011 0001 1011

= 0111 1101 0100 1111 <<< 3

= 1110 1010 0111 1011 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1011 1110 0110 0111 + 1110 1010 0111 1011 + 0110 1011 0001 1011

= 0001 0011 1111 1101 <<< 6

= 1111 1111 0100 0100 i = (11 + 1) % 14 = 12

j = (1 + 1) % 2 = 0

do 41:

A = S[12] = (S[12] + A + B) <<< 3

= 0100 0010 0111 0010 + 1110 1010 0111 1011 + 1111 1111 0100 0100

= 0010 1100 0011 0001 <<< 3

= 0110 0001 1000 1001 B = L[0] = (L[0] + A + B) <<< ((A + B)%16)

= 0110 1011 0001 1011 + 0110 0001 1000 1001 + 1111 1111 0100 0100

= 1100 1011 1110 1000 <<< 13

= 0001 1001 0111 1101

i = (12 + 1) % 14 = 13

j = (0 + 1) % 2 = 1

do 42:

A = S[13] = (S[13] + A + B) <<< 3

= 0001 0001 0111 0001 + 0110 0001 1000 1001 + 0001 1001 0111 1101

= 1000 1100 0111 0111 <<< 3

= 0110 0011 1011 1100 B = L[1] = (L[1] + A + B) <<< ((A + B)%16)

= 1111 1111 0100 0100 + 0110 0011 1011 1100 + 0001 1001 0111 1101

= 0111 1100 0111 1101 <<< 9

= 1111 1010 1111 1000

i = (13 + 1) % 14 = 0

j = (1 + 1) % 2 = 0

3. Setelah melakukan seluruh proses *key expansion* proses selanjutnya adalah melakukan proses algoritma CBC mode enkripsi

   Data = 0037 = 0000 0000 0011 0111

   A = Data (0,8) = 0000 0000 -> 16 bits = 0000 0000 0000 0000

   B = Data (8,16) = 0011 0111 -> 16 bits = 0000 0000 0011 0111

   A = A XOR IV1

   = 0000 0000 0000 0000 XOR 0000 0000 0001 0010

   = 0000 0000 0001 0010

   B = B XOR IV2

   = 0000 0000 0011 0111 XOR 0000 0000 0001 0010

   = 0000 0000 0010 0101

4. Selanjutnya adalah proses enkripsi menggunakan algoritma RC5

   A = A + S[0]

   = 0000 0000 0001 0010 + 1010 0100 1100 1000 = 1010 0100 1101 1010

   B = B + S[1]

   = 0000 0000 0010 0101 + 1111 1011 1110 1010 = 1111 1100 0000 1111

   For i = 1 to 6 do :

   A = ((A XOR B) <<< B) + S[2*i]

   B = ((B XOR A) <<< A) + S[2*i+1]

   Dari algoritmadi atas proses perulangan yang dilakukan adalah sebagai berikut

   Do 1 :

   A = ((A XOR B) <<< 15) + S[2]

   = 1010 0100 1101 1010 XOR 1111 1100 0000 1111

= 0101 1000 1101 0101 <<< 15

= 1010 1100 0110 1010 + 1101 0110 1001 0101

= 1000 0010 1111 1111

B = ((B XOR A) <<< 15) + S[3]

= 1111 1100 0000 1111 XOR 1000 0010 1111 1111

= 0111 1110 1111 0000 <<< 15

= 0011 1111 0111 1000 + 0001 1101 1111 1010

= 0101 1101 0111 0010

Do 2 :

A = ((A XOR B) <<< 2) + S[4]

= 1000 0010 1111 1111 XOR 0101 1101 0111 0010

= 1101 1111 1000 1101 <<< 2

= 0111 1110 0011 0111 + 1111 0101 0001 1010

= 0111 0011 0101 0001

B = ((B XOR A) <<< 1) + S[5]

= 0101 1101 0111 0010 XOR 0111 0011 0101 0001

= 0010 1110 0010 0011 <<< 1

= 0101 1100 0100 0110 + 1000 0011 0001 0011

= 1101 1111 0101 1001

Do 3 :

A = ((A XOR B) <<< 9) + S[6]

= 0111 0011 0101 0001 XOR 1101 1111 0101 1001

= 1010 1100 0000 1000 <<< 9

= 0001 0001 0101 1000 + 0100 1001 0011 0010

= 0101 1010 1000 1010

B = ((B XOR A) <<< 10) + S[7]

= 1101 1111 0101 1001 XOR 0101 1010 1000 1010

= 1000 0101 1101 0011 <<< 10

= 0100 1110 0001 0111 + 1111 0111 0100 1000

= 0100 0101 0101 1111

Do 4 :

A = ((A XOR B) <<< 15) + S[8]

= 0101 1010 1000 1010 XOR 0100 0101 0101 1111

= 0001 1111 1101 0101 <<< 15

= 1000 1111 1110 1010 + 1111 0011 0110 1111

= 1000 0011 0101 1001

B = ((B XOR A) <<< 9) + S[9]

= 0100 0101 0101 1111 XOR 1000 0011 0101 1001

= 1100 0110 0000 0110 <<< 9

= 0000 1101 1000 1100 + 0100 1010 0110 0101

= 0101 0111 1111 0001

Do 5 :

A = ((AXORB) <<< 1) + S[10]

= 1000 0011 0101 1001 XOR 0101 0111 1111 0001

= 1101 0100 1010 1000 <<< 1

= 1010 1001 0101 0001 + 0001 0000 0110 0001

= 1011 1001 1011 0010

B = ((BXORA) <<< 2) + S[11]

= 0101 0111 1111 0001 XOR 1011 1001 1011 0010

= 1110 1110 0100 0011 <<< 2

= 1011 1001 0000 1111 + 1110 1010 0111 1011

= 1010 0011 1000 1010

Do 6 :

A = ((AXORB) <<< 10) + S[12]

= 1011 1001 1011 0010 XOR 1010 0011 1000 1010

= 0001101000111000 <<< 10

= 1110 0000 0110 1000 + 0110 0001 1000 1001

= 0100 0001 1111 0001

B = ((BXORA) <<< 1) + S[13]

= 1010 0011 1000 1010 XOR 0100 0001 1111 0001

= 1110 0010 0111 1011 <<< 1

= 1100 0100 1111 0111 + 0110 0011 1011 1100

= 0010 1000 1011 0011

5. Setelah melakukan semua proses, maka menghasilkan *ciphertext* sebagai berikut

   CT [0] = A = 0100 0001 1111 0001 = 41F1

   CT [1] = B = 0010 1000 1011 0011 = 28B3

6.1.2 Pengujian Mekanisme Pengiriman Data *Node* Pengirim dengan *Node Gateway*

Pengujian selanjutnya adalah mekanisme pengiriman data *node* pengirim dengan *node gateway*. Mekanisme tersebut merupakan fungsi dari pengiriman data kedua *node* dengan menggunakan perangkat NRF24L01 yang ada pada *node* pengirim maupun pada *node gateway*. Pengujan ini memiliki tujuan untuk memastikan *node* pengirim dan *node gateway* dapat berkomunikasi sesuai dengan proses implementasi dan sesuai dengan perancangan.

Gambar 6.2 Pengiriman data dari *node* pengirim

Pada gambar 6.2 proses pengiriman data berhasil dilakukan, dimana data yang dikirimkan adalah data *ciphertext* yang dihasilkan dari proses enkripsi.

Gambar 6.3 Penerimaan data oleh *node gateway*

Pada gambar 6.3 proses penerimaan data berasil dilakukan, dimana data yang diterima adalah data *ciphertext* yang dikirimkan oleh *node* pengirim.

Tabel 6.2 Tabel pengujian mekanisme pengiriman data *node* pengirim dengan *node gateway*

No.	Test Name	Test Case	Expected Result	Result	Status
1.	REQ-ProTiny-senData-2000-Inisialisasi jalur radio komunikasi 2 *node*	Melakukan pengujian pada fungsi `radio.openWritingPipe()` dan `radio.openReadingPipe(`	Perangkat sensor pada *node* pengirim menginisiasi jalur dengan menggunakan fungsi `radio.openWritingP`	Perangkat sensor pada *node* pengirim menginisiasi jalur dengan menggunakan fungsi `radio.openWritingPip`	Valid

No.	Test Name	Test Case	Expected Result	Result	Status
		`)` untuk melakukan inisialisasi jalur yang komunikasi *node* pengirim dan *node gateway* digunakan	`ipe()` dan *node gateway* menggunakan fungsi `radio.openReadingPipe()` dan berhasil mengirimkan data dengan address yang sama	`e()` dan *node gateway* menggunakan fungsi `radio.openReadingPipe()` dan berhasil mengirimkan data dengan address yang sama ditandai dengan data yang dikirimkan dan diterima sama	
2.	REQ-ProTiny-senData-2001-Pengiriman data dari *node* pengirim	Melakukan pengujian pada fungsi `radio.write()` Untuk mengirim data yang disimpan pada *payload* data dikirim kepada *node gateway*	Perangkat *node* pengirim dapat mengirimkan data yang ada pada data *payload*	Perangkat *node* pengirim dapat mengirimkan data sesuai dengan data yang sesuai dengan data yang disimpan pada data *payload*	Valid
3.	REQ-ProTiny-sendData-2002-Penerimaan data oleh *node* ateway	Melakukan pengujian pada fungsi `radio.read()` untuk menerima data *payload* yang dikirimkan oleh *node* pengirim	Perangkat *node gateway* dapat menerima data *payload* sesuai dengan yang dikirimkan pada data	Perangkat *node gateway* dapat menerima data *payload* sesuai dengan yang dikirimkan pada data	Valid

No.	Test Name	Test Case	Expected Result	Result	Status
			payload node pengirim	*payload node* pengirim	

6.1.3 Pengujian Mekanisme Dekripsi Data Protokol Kemanan TinySec *Node Gateway*

Pengujian proses berikutnya adalah mekanisme dekripsi data protokol keamanan TinySec *node gateway* yang mengacu pada rekayasa kebutuhan serta implementasi yang telah dijelaskan pada bab sebelumnya. Tujuan dari pengujian ini adalah untuk melakukan verifikasi terhadap hasil implementasi yang telah dilakukan. Proses dekripsi data *ciphertext* dengan menggunakan protokol keamanan TinySec berbasis algoritma RC5 dan CBC mode yang telah dikirimkan oleh *node* pengirim dan diterima oleh *node gateway*.

Gambar 6.4 Dekripsi data pada *node gateway* telah dilakukan

Pada gambar 6.4 menggambarkan bahwa proses dekripsi telah dilakukan melalui proses penerimaan data *ciphertext* dari *node* pengirim, inisiaslisasi *key*, IV, proses ekspansi *key*, dna menghasilkan *Plaintext*

Tabel 6.3 Tabel pengujian mekanisme dekripsi data protokol keamanan TinySec *node gateway*

No.	Test Name	Test Case	Expected Result	Result	Status
1.	REQ-ProTiny-Decrypt-3000-	Melakukan pengujian perangkat	Perangkat *node gateway* dapat	Perangkat *node gateway* dapat	Valid

No.	Test Name	Test Case	Expected Result	Result	Status
	Penerimaan data *ciphertext*	lunak memiliki fungsi untuk menerima data dan menyimpan data pada sebuah variabel sebagai *ciphertext*	menerima data dari *node* pengirim berupa data *ciphertext*	menerima data dari *node* pengirim berupa data *payload* melalui komunikasi radio yang akan disimpan pada variabel *ciphertext*	
2.	REQ-ProTiny-Decrypt-3001-Insialisasi Intializatio n Vector	Melakukan pengujian perangkat lunak memiliki fungsi untuk mendapatkan atau menginisialisasi Intialization Vector (IV)	Perangkat *node gateway* dapat menginisialisasi nilai IV yang dibutuhkan sebagai parameter pada CBC mode	Perangkat *node gateway* dapat menginisialisasi nilai IV yang dibutuhkan sebagai parameter pada CBC mode dengan nilai IV pertama ditentukan sejak awal	Valid
3.	REQ-ProTiny-Decrypt-3002-Inisialisasi *key* dan ekspansi *key*	Melakukan pengujian perangkat lunak memiliki fungsi untuk menginisialisasi *key* dan melakukan ekspansi *key*	Perangkat *node gateway* dapat melakukan inisiasi *key* dan melakukan ekspansi *key* dengan total yang yang	Perangkat *node gateway* dapat melakukan inisiasi *key* awal dan melakukan ekspansi *key* hingga	Valid

No.	Test Name	Test Case	Expected Result	Result	Status
			telah ditentukan	mengahasilkan 14 sub*key*	
4.	REQ-ProTiny-Decrypt-3003-Dekripsi data	Melakukan pengujian pada fungsi `dekripsi()` untuk melakukan enkripsi data yang telah di akusisi menggunakan algoritma CBC-mode dan RC5	Perangkat *node gateway* dapat menghasilkan *plaintext* dari proses dekripsi sehingga data sensor dapat dibaca	Perangkat *node gateway* dapat menghasilkan *plaintext* dari proses dekripsi sehingga data sensor dapat dibaca dan ditampilkan	Valid

Dari hasil pengujian diatas, semua proses dalam mekanisme dekripsi data protokol keamanan *node* pengirim dinyatakan valid dan berhasil dilakukan. Jika dianalisa proses mekanisme dekripsi protokol keamanan TinySec berbasis algoritma CBC mode dekripsi tersebut dapat dijelaskan sebagi berikut.

1. Proses yang sama dilakukan pada proses enkripsi yaitu inisialisasi prameter yang dibutuhkan dan *key expansion* dilakukan dengan proses yang sama pada proses enkripsi, dan menghasilkan nilai hasil *key expansion* yang sama juga.
2. Selanjutnya melakukan proses dekripsi menggunakan dekripsi algoritma RC5

   for i = r down to 1 do:

   B = ((B - S[2 * i + 1]) >>> A) XOR A

   A = ((A - S[2 * i]) >>> B) XOR B

   B = B - S[1]

   A = A - S[0]

   Dari algoritma diatas maka menghasilkan perulang sebagai berikut

   do 6 :

   B = ((B-S[13]>>>1) XOR A

   = 0010 1000 1011 0011 - 0110 0011 1011 1100

   = 1100 0100 1111 0111 >>> 1

   = 1110 0010 0111 1011 XOR 0100 0001 1111 0001

   = 1010 0011 1000 1010

   A = ((A-S[12]>>>10) XOR B

= 0100 0001 1111 0001 - 0110 0001 1000 1001

= 1110 0000 0110 1000 >>> 10

= 0001 1010 0011 1000 XOR 1010 0011 1000 1010

= 1011 1001 1011 0010

do 5 :

B = ((B-S[11]>>>2) XOR A

= 1010 0011 1000 1010 - 1110 1010 0111 1011

= 1011 1001 0000 1111 >>> 2

= 1110 1110 0100 0011 XOR 1011 1001 1011 0010

= 0101 0111 1111 0001 A = ((A-S[10]>>>1) XOR B

= 1011 1001 1011 0010 - 1011 1001 1011 0010

= 1010 1001 0101 0001 >>> 1

= 1101 0100 1010 1000 XOR 0101 0111 1111 0001

= 1000 0011 0101 1001

do 4 :

B = ((B-S[9]>>>9) XOR A

= 0101 0111 1111 0001 - 0100 1010 0110 0101

= 0000 1101 1000 1100 >>> 9

= 1100 0110 0000 0110 XOR 1000 0011 0101 1001

= 0100 0101 0101 1111

A = ((A-S[8]>>>15) XOR B

= 1000 0011 0101 1001 - 1000 0011 0101 1001

= 1000 1111 1110 1010 >>> 15

= 0001 1111 1101 0101 XOR 0100 0101 0101 1111

= 0101 1010 1000 1010

Do 3 :

B = ((B-S[7]>>>10) XOR A

= 0100 0101 0101 1111 - 1111 0111 0100 1000

= 0100 1110 0001 0111 >>> 10

= 1000 0101 1101 0011 XOR 0101 1010 1000 1010

= 1101 1111 0101 1001

A = ((A-S[6]>>>9) XOR B

= 0101 1010 1000 1010 - 0100 1001 0011 0010

= 0001 0001 0101 1000 >>> 9

= 1010 1100 0000 1000 XOR 11011 1110 101 1001

= 0111 0011 0101 0001

Do 2 :

B = ((B-S[5]>>>1) XOR A

= 1101 1111 0101 1001 - 1000 0011 0001 0011

= 0101 1100 0100 0110 >>> 1

= 0010 1110 0010 0011 XOR 0111 0011 0101 0001

= 0101 1101 0111 0010

A = ((A-S[4]>>>2) XOR B

= 0111 0011 0101 0001 - 1111 0101 0001 1010

= 0111 1110 0011 0111 >>> 2

= 1101 1111 1000 1101 XOR 0101 1101 0111 0010

= 1000 0010 1111 1111

Do 1 :

B = ((B-S[3]>>>15) XOR A

= 0101 1101 0111 0010 - 0001 1101 1111 1010

= 0011 1111 0111 1000 >>> 15

= 0111 1110 1111 0000 XOR 1000 0010 1111 1111

= 1111 1100 0000 1111

A = ((A-S[2]>>>15) XOR B

= 1000 0010 1111 1111 - 1101 0110 1001 0101

= 1010 1100 0110 1010 >>> 15

= 0101 1000 1101 0101 XOR 1111 1100 0000 1111

= 1111 1100 0000 1111

A = A - S[0]

= 1111 1100 0000 1111 + 1010 0100 1100 1000 = 0000 0000 0001 0010

B = B - S[1]

= 1111 1100 0000 1111 + 1111 1011 1110 1010 = 0000 0000 0010 0101

3. Proses terakhir yang dilakukan adalah melakukan proses dekripsi CBC mode

   A = A ^ IV1

   = 0000 0000 0001 0010 ^ 0000 0000 0001 0010

   = 0000 0000 0000 0000

   B = B ^ IV2

   = 0000 0000 0010 0101 ^ 0000 0000 0001 0010

   = 0000 0000 0011 0111

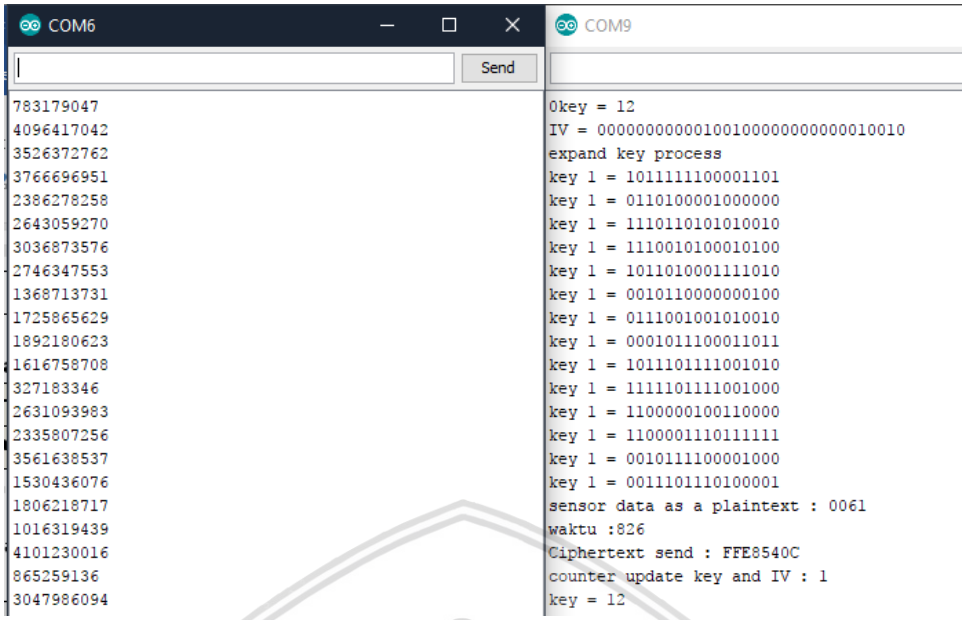
   Data = A(8,16) + B (8,16)

   = 0000 0000 0011 0111

   = 0037

6.1.4 Pengujian Mekanisme *Listening* Data *Ciphertext* pada *Node Attacker*

Pengujian selanjutnya adalah mekanisme *Listening* data *ciphertext* pada *node attacker*. Mekanisme tersebut merupakan fungsi dari pengiriman data kedua *node* dengan menggunakan perangkat NRF24L01 yang ada pada *node* pengirim maupun pada *node attacker*. Pengujian ini memiliki tujuan untuk memastikan *node* pengirim dan *node attacker* dapat berkomunikasi sesuai dengan proses implementasi dan sesuai dengan perancangan.



Gambar 6.5 Proses *listening* data *ciphertext* pada *node attacker* dan pengiriman data oleh *node* pengirim

Pada gambar 6.5 menggambarkan proses *listening* data yang dilakukan oleh *node attacker* mencoba untuk mendapatkan data dari proses komunikasi *wireless* menggunakan NRF24L01 pengiriman yang dilakukan secara *broadcast*, mudah untuk di serang.

Tabel 6.4 Tabel pengujian mekanisme *node* data *ciphertext* pada *node attacker*

No.	Test Name	Test Case	Expected Result	Result	Status
1.	REQ-ProTiny-*listen*Atck-4000-Inisialisasi jalur radio komunikasi 2 *node*	Melakukan pengujian pada fungsi `radio.openWritingPipe()` dan `radio.openReadingPipe()` untuk melakukan inisialisasi jalur yang komunikasi *node* pengirim dan *node*	Perangkat sensor pada *node* pengirim menginisiasi jalur dengan menggunakan fungsi `radio.openWritingPipe()` dan *node attacker* menggunakan fungsi `radio.openReadingPipe()` dan	Perangkat sensor pada *node* pengirim menginisiasi jalur dengan menggunakan fungsi `radio.openWritingPipe()` dan *node attacker* menggunakan fungsi `radio.openReadingPipe()` dan berhasil	Valid

No.	Test Name	Test Case	Expected Result	Result	Status
		attacker digunakan	berhasil mengirimkan data dengan address yang sama	mengirimkan data dengan address yang sama ditandai dengan data yang dikirimkan dan diterima sama	
2.	REQ-ProTiny-*listen*Atck-4001-Pengiriman data dari *node* pengirim	Melakukan pengujian pada fungsi `radio.write()` Untuk mengirim data yang disimpan pada *payload* data dikirim kepada *node attacker*	Perangkat *node* pengirim dapat mengirimkan data yang ada pada data *payload*	Perangkat *node* pengirim dapat mengirimkan data sesuai dengan data yang sesuai dengan data yang disimpan pada data *payload*	Valid
3.	REQ-ProTiny-*listen*Atck-4002-Penerimaan data oleh *node* ateway	Melakukan pengujian pada fungsi `radio.read()` untuk menerima data *payload* yang dikirimkan oleh *node* pengirim	Perangkat *node attacker* dapat menerima data *payload* sesuai dengan yang dikirimkan pada data *payload node* pengirim	Perangkat *node attacker* dapat menerima data *payload* sesuai dengan yang dikirimkan pada data *payload node* pengirim	Valid
4.	REQ-ProTiny-*listen*Atck-4003-	Melakukan pengujian pada fungsi `Serial.print()` untuk	Perangkat *node attacker* dapat	Perangkat *node attacker* dapat menampilkan	Valid

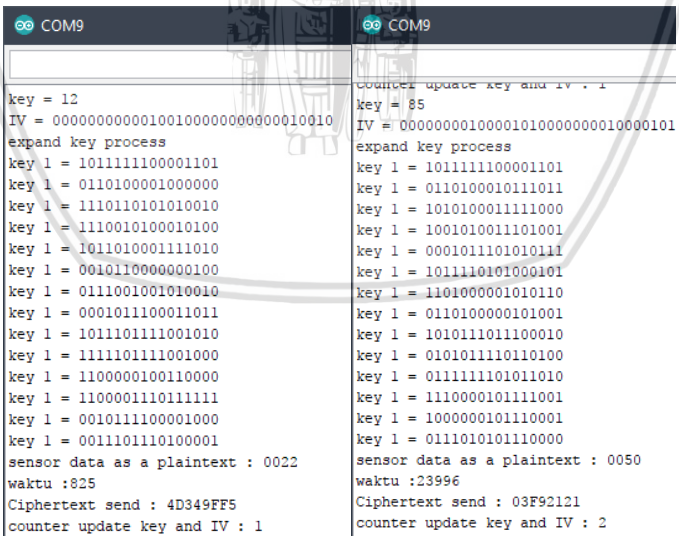
No.	Test Name	Test Case	Expected Result	Result	Status
	Pembacaan data	menampilkan data *ciphertext* yang berasil di *listen* oleh *node attacker*	menampilkan data hasil *listening* terhadap *node* pengirim	data hasil *listening* terhadap *node* pengirim menggunakan `serial.print()`	

6.2 Pengujian Kebutuhan Non-Fungsional

Pengujian untuk memnuhi kebutuhan non-fungsional dilakukan menyesuaikan dengan yang telah disebutkan pada rekayasa kebutuhan yang meliputi kebutuhan penigkatan performa *key* dan IV.

6.2.1 Pengujian Kebutuhan Peningkatan Performa *Key* dan IV

Pengujian selanjutnya adalah pengujian kebutuhan peningkatan performa *key* dan IV, tujuan pengujian ini untuk memenuhi rekayasa kebutuan dan menyesuaikan implementasi pada bab sebelumya. Kondisi *key* dan IV yang static tidak berpengaruh terhadap algoritma yang diterapkan, namun berpengaruh dalam performa pengamanan data.



Gambar 6.6 Proses perubahan *key* dan IV pada data ke 11 setelah *counter* ke 10

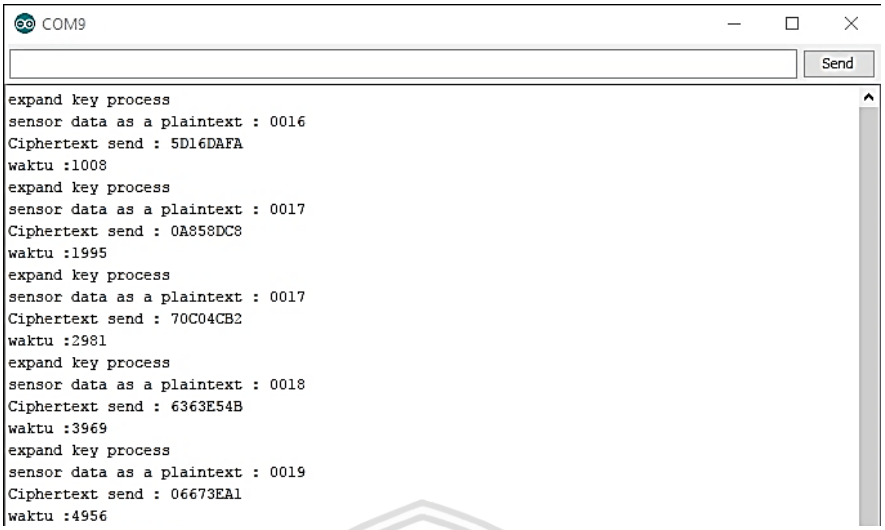
Pada gambar 6.5 menampilkan proses perubahan *key* dan IV pada saat *counter* ke 10, atau proses pengiriman dan penerimaan data pada proses ke 11. *Key* akan terus berubah secara berkala ketika *counter* menunjukan nilai = 10.

Tabel 6.5 Tabel pengujian mekanisme dekripsi data protokol keamanan TinySec *node gateway*

No.	Test Name	Test Case	Expected Result	Result	Status
1.	REQ-ProTiny-NF-2000-*Update key* dan IV	Melakukan proses penerimaan dan pengiriman data selama 10 kali iterasi selanjutnya *key* dan IV akan di inisialisasi kembali dengan menggunakan fungsi `random()`	Perangkat *node* pengirim dan *gateway* dapat melakukan *update key* dan IV setiap proses pengiriman dan penerimaan data kelipatan 10 dengan menggunakan fungsi `random()`	Perangkat *node* pengirim dan *gateway* dapat melakukan *update key* dan IV setiap proses pengiriman dan penerimaan data kelipatan 10 dengan menggunakan fungsi `random()`	Valid

6.2.2 Pengujian Performansi Kecepatan Enkripsi dan Dekripsi

Pengujian kecepatan proses enkripsi dan dekripsi memiliki tujuan untuk mengetahui tingkat performansi kecepatan penerapan protokol keamanan TinySec dalam proses enkripsi dan dekripsi. Proses melakukan pengujian adalah dengan melakukan proses enkripsi dan dekripsi selama 5 kali untuk mendapatkan waktu proses yang dibutuhkan dengan menggunakan fungsi `milis()` yang ada pada Arduino .



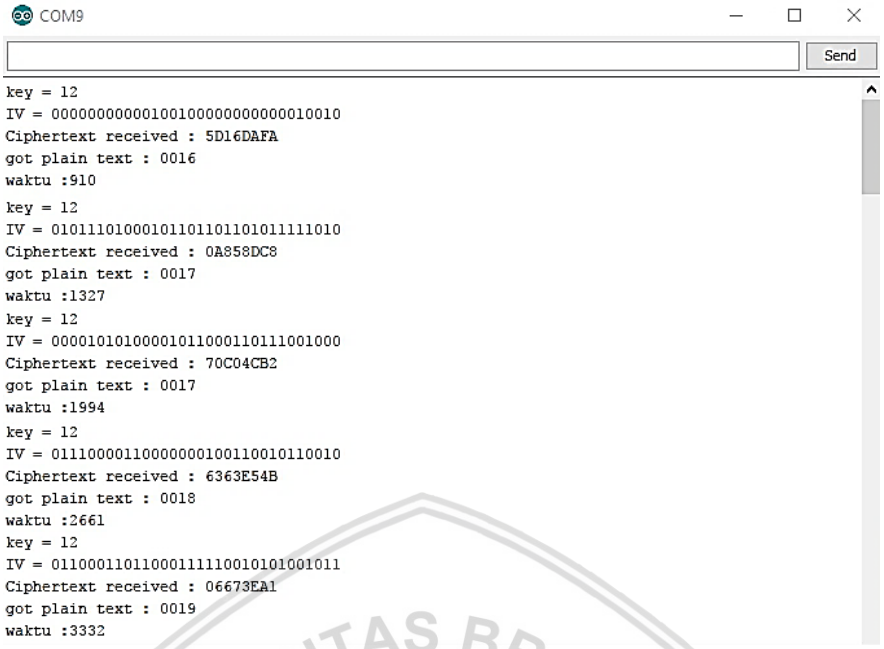
Gambar 6.7 Tampilan hasil enkripsi dan waktu proses yang dibutuhkan

Pada gambar 6.7 merupakan tampilan hasil enkripsi yang dilakukan selama 5 kali proses, dengan tambahan parameter waktu proses, sehingga jika dilihat dari waktu proses yang di dapatkan dari fungsi milis(), menghasilkan nilai *timestamp* pada tiap prosesnya. Untuk mendapatkan waktu tiap proses dapat melihat selisih tiap *timestamp*nya, hasil tersebut dapat dilihat pada Tabel 6.6 berikut.

Tabel 6.6 Tabel hasil percobaan dan selisih waktu proses enkripsi

Percobaan ke-	Plaintext	*Ciphertext*	Waktu proses (ms)
1	0016	5D16DAFA	1008
2	0017	0A858DC8	987
3	0017	70C04CB2	986
4	0018	6363E54B	988
5	0019	06673EA1	987
Nilai rata – rata waktu proses enkripsi			991

Pada Tabel 6.6 tiap proses memiliki waktu yang cukup konstan, dengan rata-rata waktu proses yang dibutuhkan selama 5 kali proses tersebut adalah 991 ms. Total waktu yang dibutuhkan dalam proses enkripsi, merupakan total waktu yang dimiliki oleh *node* pengirim, dengan proses akuisisi data dan proses pengiriman data menuju *node gateway*.



Gambar 6.8 Tampilan hasil dekripsi dan waktu prposes yang dibutuhkan

Pada gambar 6.8 merupakan tampilan hasil dekripsi yang dilakukan selama 5 kali proses, dengan tambahan parameter waktu proses, sehingga jika dilihat dari waktu proses yang sisapatkan dari fungsi milis, menghasilkan nilai *timestamp* pada tiap prosesnya. Untuk mendapatkan waktu tiap proses dpat melihat selisih tiap *timestamp*nya, hasil tersebut dapat dilihat pada tabel 6.7 berikut.

Tabel 6.7 Tabel hasil percobaan dan selisih waktu proses dekripsi

Percobaan ke-	*Ciphertext*	Plaintext	Waktu proses (ms)
1	5D16DAFA	0016	910
2	0A858DC8	0017	417
3	70C04CB2	0017	667
4	6363E54B	0018	667
5	06673EA1	0019	671
Nilai rata – rata waktu proses dekripsi			666

Pada Tabel 6.7 berbeda dengan proses enkripsi, perbedaan waktu proses dekripsi pada percobaan 1, 2 dan 3 memiliki perbedaan yang cukup signifikan. Dimana hal ini juga dipengaruhi oleh proses penerimaan data dari *node* pengirim oleh *node gateway*. Perhitungan yang di jalan kan oleh fungsi `milis()` terus berjalan dimulai dari saat mikrokontroler menyala.

BAB 7 PENUTUP

Pada bab ini membahas kesimpulan diperoleh dari penelitian ini, serta saran-saran untuk pengembangan penelitian ini lebih lanjut.

7.1 Kesimpulan

Berdasarkan rumusan masalah yang dibuat serta hasil perancanga, implementasi dan pengujian pada penelitian ini, makan dapat diambil kesimpulan sebagai berikut.

1. Protokol keamanan TinySec memiliki beberapa memiliki dua mode utama yaitu, autheticated dan authencticated *encryption*. Secara keseluruhan telah memenuhi kebutuhan security goals, yaitu *confidentiality, Authetication, dan Integrity*. Namun pada penelitian kali ini hanya menerapkan mode *autheticated encryption* dengan memenuhi *security goals confidentiality*. Keterbatasan yang dimiliki perangkat mikrokontroler berbasis ATmega328p menjadi permasalahan dalam penerapan protokol keamanan TinySec, begitu juga dengan keterbatasan media pengiriman NRF24L01 melalui frekuensi radio. Proses penerapan protokol keamanan TinySec berbasis algoritma RC5 dan algoritma CBC-mode tetap bisa diterapkan pada mikrokontroler berbasis ATmega328p, dengan beberapa batasan seperti, jumlah *word*, total *key*, dan total *round.*
2. Perubahan tingkat performansi pada *Wireless Sensor Network* setelah menerapkan protokol keamanan TinySec terjadi pada tingkat performa kecepatan proses data. dimana kecepatan proses data untuk melakukan proses enkripsi membutuhkan waktu hingga 991 ms dengan total proses data yang dilakukan adalah akuisisi data sensor, enkripsi data sensor, dan pengiriman data melalui radio frekuensi. Untuk melakukan proses dekripsi membutuhkan waktu hingga 666 ms dengan total proses data yang dilakukan adalah menerima data yang dikirimkan dan melakukan proses dekripsi. Perubahan tingkat performansi ini terjadi karena proses enkripsi dan dekripsi yang cukup banyak, namun dalam segi keamanan data tentu data yang dikirimkan melalui radio frekuensi akan lebih aman setelah melakukan proses enkripsi menggunakan protokol keamanan TinySec, hal tersebut dibuktikan dengan proses pengujian listening data oleh *node attacker* hanya mendapatkan data berupa *ciphertext* saja, sehingga data asli yang dikirimkan oleh *node* pengirim dan diterima oleh *node gateway* tidak didapatkan oleh *node attacker*.

7.2 Saran

Adapun beberapa saran yang dapat diberikan untuk pengembangan penelitian selanjutnya sebagai berikut :

1. Disarankan mengetahui kebutuhan protokol keamanan yang akan diterapkan dan spesifikasi yang dimiliki mikrokontroler, sehingga dapat memaksimalkan fungsi yang dimiliki protokol keamanan yang ingin diterapkan.
2. Penelitian ini dapat dikembangkan kembali dalam hal perubahan mikrokontroler yang lebih sesuai dengan kebutuhan protokol keamanan TinySec, media komunikasi data baik secara satu arah maupun dua arah.

DAFTAR PUSTAKA

Danuansa, R., Yulianto, F. A. & Prabowo, S., 2017. *Performansi dan Simulasi Security Protocol TinySec dan LLSP pada Wireless.* Bandung, Universitas Telkom.

Dr Xuemin Shen, D. Y. P., 2010. *Fundamental of Wireless Sensor Network, series wiley series on wireless communications and mobile computing.* singapore: markono.

Evans, D., 2011. The Internet of Things, How the Next Evolution of the Internet Is Changing Everything. *White Paper,* pp. 2-4.

Huang, K.-T., Chiu, J.-H. & Shen, S.-S., 2013. *A NOVEL STRUCTURE WITH DYNAMC OPERATION MODE FOR SYMMETRIC-KEY BLOCK CHIPERS.* Taipei, International Journal of Network Security & Its Applications (IJNSA).

Hutomo, A. B. A., S. & Prabowo, S., 2017. *Impelentasi dan Analisis Perbandingan Performa Wireless Sensor Network Security Protocol.* Bandung, Universitas Telkom.

Laubhan, K. et al., 2016. A Four-Layer Wireless Sensor Network Framework. p. 2.

Leonardo, A., Akbar, S. R. & Maulana, R., 2017. *Implementasi Smart Light Bulb Pervasive Berbasis ESP8266.*

Lighfoot, L. E., Ren, J. & Li, T., 2007. *An Energy Efficient Link-Layer Security Protocol for Wireless Sensor Network.* s.l., s.n., p. 233.

Ren, J., Li, T. & Aslam, D., 2005. A POWER EFFICIENT LINK-LAYER SECURITY PROTOCOL (LLSP) FOR WIRELESS SENSOR NETWORKS. p. 2.

Rivest, R. L., 1997. *The RC5 Encryption Algorithm.* Cambridge, MIT Laboratory fir Computer Science.

Sen, J., 2009. A survey on Wireless Sensor Network Security. p. 1.

Shobrina, U. J., Primananda, R. & Maulana, R., 2016. *Analisis Kinerja Pengiriman Data Modul Transceiver NRF24l01, Xbee dan Wifi ESP8266 Pada Wireless Sensor Network,* Volume II, p. 3.